UNIVERSIDAD REY JUAN CARLOS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA INFORMÁTICA

Curso Académico 2009/2010

Proyecto de Fin de Carrera

SISTEMA DE GESTIÓN Y CORRELACIÓN DE EVENTOS DE SEGURIDAD EN SISTEMAS CRÍTICOS

Alumno: Nicolás Moreno Ishii

Tutor: Rafael Capilla Sevilla
Cotutor: Francisco Estaire Estaire (UPM)

Dedicado a mi familia y en especial a mis padres

Agradecimientos

Es fácil olvidarse de gente que ha estado contigo a lo largo de todos estos años, que te han ayudado en un momento concreto y te han empujado a conseguir aquello que tanto buscabas. A todos ellos, gracias.

El hecho de estudiar una ingeniería, ha cambiado mi vida. Surge en mí un interés muy especial, que quizás nunca hubiera sentido, por las cosas que me rodean. Te vuelves más observador, detallista, metódico y crítico. Intentas comprender cómo funciona aquello que te llama realmente la atención.

Durante este camino, he vivido mil y una anécdotas, mil y una experiencias, tanto positivas, como negativas, que me han hecho madurar y formarme como persona. Me gustaría agradecer a todos los profesores que he tenido en la carrera, su dedicación y profesionalidad. Todos han aportado algo en mi educación, unos más y otros obviamente menos.

Quiero agradecer a Rafael Capilla, por su disposición, accesibilidad, dirección y consejos, a lo largo de la carrera y en especial este último año. Sin él, no hubiera conocido a Francisco Estaire, al que admiro personal y profesionalmente, por darme la oportunidad de trabajar con él y por la confianza que ha depositado en mí.

En el plano personal, agradecer a mis compañeros de viaje de todos estos años. Sin tener en cuenta el orden, agradecer a Javi, José, Blasco, Jorge, Moreno, Álvaro y Oscar. Además de Jesús y Miguel, que siempre han estado presentes cuando les he necesitado. Han sido y serán mi mejor apoyo.

Quiero acordarme de mi familia, en el que incluyo mis amigos más íntimos. A Borja, Roberto y Ángel, como mis mejores amigos, siempre al pie del cañón. Agradecerles por su ánimo y sobre todo, su compresión en estos largos años. Finalmente, dar gracias a mi familia, a mi padre Luís, por transmitirme la importancia de la pasión, constancia y amor propio, en cualquier aspecto de la vida. A mi madre Afi, por enseñarme a ser paciente y reflexivo. Y a mi hermana Alicia, por hacerme creer que los retos y los sueños pueden cumplirse, si realmente lo deseas.

Índice

Índice de ilustraciones

Índice de tablas

Resumen

Durante los últimos años se ha aumentado las medidas de protección de los sistemas en las organizaciones con el fin de defenderse de ataques internos y externos a las instituciones, incrementando considerablemente la necesidad de gestionar de forma adecuada la seguridad de la información.

La necesidad de saber qué está ocurriendo y qué puede ocurrir en nuestros sistemas críticos ante ataques maliciosos, requiere que la gestión de la seguridad sea vista desde la perspectiva de la eficiencia y mejora continua. Por ello, la monitorización de los sistemas informáticos tiene como objetivo la detección de posibles vulnerabilidades, amenazas, a través de la gestión de los eventos de seguridad con el fin de minimizar cualquier riesgo de ataque.

Todos los sistemas, herramientas, aplicaciones críticas y no críticas necesitan a su vez poder ser gestionadas, y facilitar así, una respuesta inmediata ante una posible ocurrencia de un incidente de seguridad. En las operaciones diarias sobre los sistemas que forman parte de la organización, generan un volumen de información valiosa relativa a su funcionamiento correcto e incorrecto. Debido a sus múltiples formatos, su gestión se hace compleja, por lo que es necesario aplicar complejos mecanismos de correlación. La correlación de eventos nos permite obtener información precisa sobre los posibles incidentes de seguridad y amenazas hacia nuestros sistemas. La respuesta en forma de alarmas, puede desencadenar un plan de acción proactiva e inteligente evitando el daño a la organización en tiempo real. A su vez, es posible emitir avisos de forma inmediata a través de alarmas, comunicando las causas de la incidencia, con la finalidad de establecer una resolución adecuada de la situación.

Cuando acontece un incidente crítico en nuestros sistemas, puede desencadenar un periodo de incertidumbre, debido a la no disposición de la información sobre las causas y posibles efectos provocados por el ataque. El análisis de los eventos y su consiguiente correlación permite reducir el número de falsos positivos e identificar en la mayor brevedad los cambios de estado y alteraciones de los comportamientos anómalos en los sistema monitorizados.

Teniendo en cuenta todas estas necesidades, se propone un sistema centralizado de gestión y correlación de eventos de seguridad en sistemas críticos. Se ha diseñado e implementado una aplicación que cubre la necesidad de gestionar todos los registros sensibles generados por aquellos dispositivos o componentes críticos dentro una organización, su almacenamiento, y el uso de motores de correlación que ayuden a prevenir y responder ante ataques maliciosos en tiempo real. Este tipo de sistemas, suelen ser habituales en medianas y grandes empresas u organismos públicos, así como entidades financieras, organismos militares, grandes empresas de ISPs, operadores dedicados al sector de las telecomunicaciones, entre otras.

Capítulo 1
Introducción

1.1 Motivación

En la actualidad nos encontramos con la necesidad de controlar o monitorizar los eventos ocurridos en un determinado sistema informático, con la finalidad de responder ante una determinada acción maliciosa.

Todos los dispositivos y aplicaciones generan una gran cantidad de información, que en muchos casos es posible recoger en forma de logs. Estos logs son creados a través de miles de eventos que ocurren durante la conexión de dispositivos o la ejecución de aplicaciones. En este sentido, la gestión de una gran cantidad de logs puede resultar compleja de procesar, por lo que resulta necesario aplicar técnicas automatizadas de gestión de eventos que nos ayuden a discriminar los distintos tipos de eventos. Una de estas técnicas es la correlación para eventos de seguridad. La necesidad de relacionar dos o más eventos independientes, en un contexto determinado, da lugar a poder reaccionar en tiempo real ante acciones o ataques maliciosos durante la monitorización del sistema comprometido.

Las técnicas de correlación de eventos son parte fundamental de los sistemas de detección de intrusiones, los cuales tratan de buscar patrones previamente definidos que impliquen cualquier tipo de actividad sospechosa o maliciosa sobre cualquier dispositivo o máquina monitorizada. Como resultado de la evaluación de los eventos, y a través de una valoración de riesgos previamente definidos, se podrían generar alertas que nos advertirán de una posible vulnerabilidad del sistema monitorizado.

El análisis de mecanismos de seguridad de los diferentes tipos de sistemas de detección de intrusiones, así como del estudio de la generación, publicación y procesamiento de eventos complejos que forman parte en las técnicas de correlación de eventos, ofrecen un campo de aplicación amplio.

Para poder entender mejor la utilidad que supone ofrecer un sistema de gestión y correlación de eventos en tiempo real, es necesario realizar un estudio de las tecnologías y paradigmas que ayudan a proteger de intrusiones indebidas de los sistemas, así como sus ventajas e inconvenientes en su uso. La idea de ofrecer un sistema que integre la gestión y almacenamiento de eventos, con mecanismos de correlación, es necesario un estudio amplio de sus tecnologías y técnicas, que se comentarán a lo largo del documento.

1.2 Objetivos

En base a los motivos expuestos anteriormente, los objetivos que pretende este proyecto son los siguientes:

- Estudio de los distintos tipos de sistemas de detección de intrusiones, analizando sus ventajas e inconvenientes desde el punto de vista de la gestión de eventos de seguridad.
- Estudio de la publicación y procesamiento de diferentes tipos de eventos, así como en la generación de ficheros de logs.
- Análisis e implementación de una aplicación que monitorice eventos generados por sistemas críticos, mediante correlación de eventos de tipo Syslog.
- Generación de respuestas en tiempo real, a través de alertas con el objetivo de evitar acciones maliciosas en el sistema monitorizado.

1.3 Método de trabajo

El método de trabajo que vamos a seguir en este proyecto es el del modelo clásico del ciclo de vida que cuenta con las siguientes fases:

a) Descripción del problema.
b) Análisis y especificación de requisitos
c) Diseño de la arquitectura software del sistema.
d) Implementación.
e) Pruebas del sistema.

Capítulo 2
Estado del arte

En esta sección estudiamos las tecnologías más importantes que intervienen en la gestión de seguridad de sistemas informáticos, con especial énfasis en la correlación de eventos.

2.1 Introducción a la seguridad informática

La necesidad de mantener un correcto funcionamiento de los sistemas informáticos provoca la necesidad de implementar mecanismos de protección que reduzcan al mínimo los riesgos asociados a la seguridad. La seguridad implica proteger una o varias entidades frente a un conjunto de riesgos en los sistemas de información. La protección de un sistema informático, implica proteger tanto los recursos hardware, como software.

La seguridad informática trata de disminuir los riesgos relacionados con el acceso y utilización de un sistema de una forma malintencionada o en ocasiones, de forma no autorizada. En este sentido, resulta adecuado implantar medidas correctivas y sobre todo preventivas, que ayuden a minimizar los riesgos de vulnerabilidades. Combinando las medidas preventivas y las correctivas, se podría considerar que el sistema a proteger, posee un nivel de seguridad adecuado.

Aún disponiendo de un plan de seguridad informática adecuado, existe la posibilidad de que nuestros datos críticos sufran daños. Los costes y beneficios de la seguridad deben ser proporcionales al valor de la información que queremos proteger.

2.1.1 Tipos de ataques a la seguridad

Un sistema libre de peligro, daño o riesgo, se entiende como aquel sistema que está exento de todo aquello que pueda afectar su funcionamiento correcto o los resultados que se obtienen del mismo. No existen sistemas totalmente seguros, por lo que es necesario tener en cuenta los posibles ataques que se pueden producir, e intentar proteger nuestros sistemas de ellos.

La clasificación de los diferentes tipos de ataques a la seguridad, se pueden agrupar en:

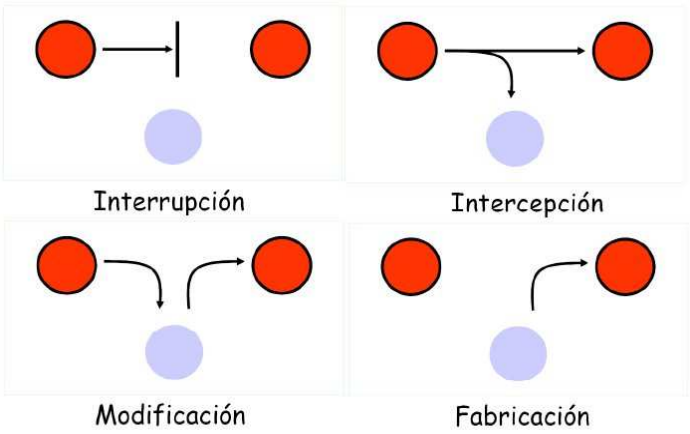


Ilustración 1 Tipos de ataques a la información

Un ataque por **interrupción**, afecta a la disponibilidad del servicio, destruyendo la información que se desea transmitir o simplemente la deja inutilizada. En este tipo de ataque, podemos mencionar un ataque por saturación a la capacidad del procesador de un determinado servidor. A este tipo de ataques se denomina "denial of service attack" (DoS attack) o ataque por denegación de servicio.

Una **interceptación**, hace referencia a una violación de la privacidad de los datos transmitidos, puede ser que una entidad no autorizada pueda obtener acceso a una información o acceso a un bien. Este ataque quedaría representado por la acción de un pinchazo o interceptación telefónica.

La **modificación** de la información transmitida influye directamente en la integridad de los datos que se quiere transmitir. Una parte no autorizada realiza una modificación sobre un recurso. Un ejemplo sería la acción de falsear datos de una transferencia bancaria o modificar, eliminar ciertas líneas de los logs de una máquina, para una futura auditoria.

Un ataque por **fabricación**, afecta a la autenticidad de la información. Por ejemplo, la posibilidad de crear eventos falsos de dispositivos que se introducen dentro del fichero de log que lo audita, provocando confusión e inutilidad en un posterior análisis **[WEB01]**.

En la ilustración 2 se muestra el crecimiento de las vulnerabilidades catalogadas por el Instituto Nacional de Tecnologías de la Comunicación (INTECO - CERT), en los últimos cuatro años. Estas vulnerabilidades han podido ser registradas gracias a informes de fuentes públicas e informes enviados directamente al INTECO - CERT **[WEB02]**, para su consiguiente análisis.

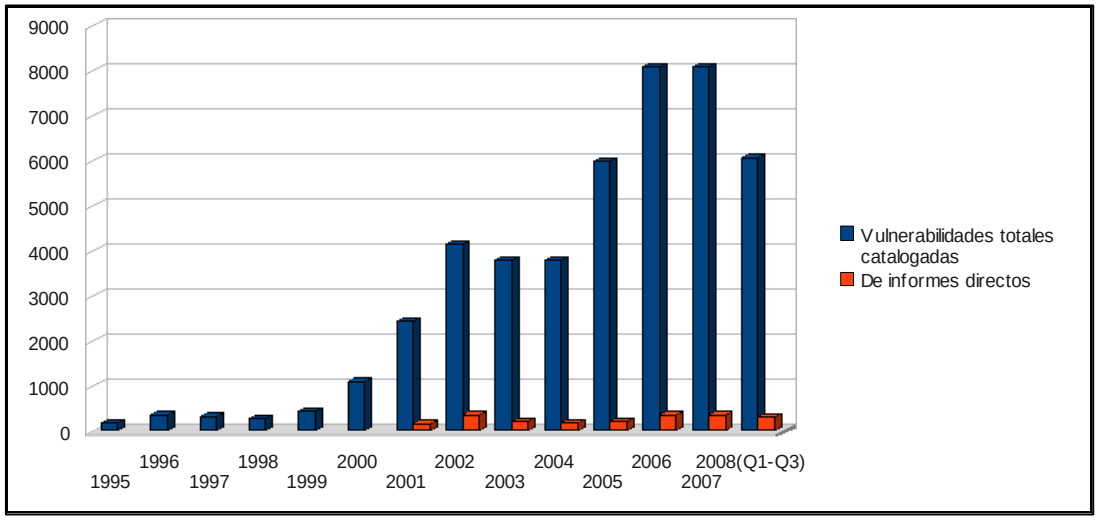


Ilustración 2 Catálogo de vulnerabilidades

Con el fin de reducir el riesgo en los sistemas informáticos, es necesario incorporar un protocolo de actuación desde un punto de vista de administrativo. La generación de políticas de seguridad y un plan de contingencia, son dos ejemplos de acción. Las políticas de seguridad, constan de normas y procedimientos internos y externos que deben seguir los empleados o responsables de la empresa, con el fin de respetar los requerimientos de seguridad que deseen preservarse.

Por otra parte, los planes de contingencia, describen una serie de pasos, que deben seguirse ante la aparición de algún evento significativo, que pueda suponer una amenaza y provocar grandes pérdidas.

2.1.2 Términos habituales relacionados con la seguridad informática

Es importante entender el verdadero significado de términos comunes cuando se habla en el ámbito de la seguridad informática. Por ello, se definen aquellos términos más utilizados en este trabajo **[WEB22]**:

- **Amenaza**: Es un evento que puede desencadenar un incidente en la organización, provocando daños materiales o pérdidas inmateriales.

- **Riesgo**: Es la probabilidad de que suceda una amenaza o evento no deseado.

- **Vulnerabilidad**: Aspectos que influyen negativamente en un recurso y que posibilita la materialización de la amenaza.

- **Ataque**: Evento con éxito o no, que atenta sobre el correcto funcionamiento del sistema.

- **Desastre o contingencia**: Interrupción de la capacidad de acceso a información y procesamiento de la misma.

2.2 Detección de intrusiones

Las intrusiones se definen como una violación de una política de seguridad establecida. Es importante tener en cuenta que es lo que está permitido y lo que no está permitido dentro de un sistema, para denominar un acceso como intrusión. El proceso de detección de intrusos en un sistema, se realiza a través de la monitorización de eventos que ocurren en una máquina o en la red, buscando las posibles señales que indiquen que se ha cometido una intrusión. Las intrusiones pueden ser causadas por atacantes de redes externas, usuarios no autorizados, o el mal uso de cierto servicio o aplicación.

Los Sistemas de Detección de Intrusiones, también llamados IDS (Intrusion Detection System), pueden estar formados por un conjunto de productos hardware o software que monitorean y analizan eventos en busca de posibles intrusiones de forma autónoma.

2.2.1 Sistemas de detección de intrusiones

Los sistemas de detección de intrusiones suelen ser infraestructuras complejas, que permiten, mediante una serie de heurísticas, detectar cuando un sistema informático está siendo utilizado de una forma inadecuada o no autorizada. Estos sistemas están constantemente supervisados, ya que incorporan mecanismos autónomos de análisis del tráfico y de sucesos del sistema, detectándolos en tiempo real. Suelen estar formados por un dispositivo hardware con una o varias interfaces, conectándose a una o varias redes. También es común la recolección de eventos en forma de logs o registros que emiten los sistemas operativos o aplicaciones críticas dentro de la propia infraestructura.

Los IDS permiten detectar ataques e intrusiones que pueden pasar desapercibidos a otros componentes que forman la seguridad del sistema. Todas las detecciones suelen ser almacenadas en forma de registros, con propósitos de reconstruir los sucesos. Podemos mencionar, que estos sistemas necesitan una administración, que suele ser en ocasiones sumamente compleja **[WEB03]**.

2.2.2 Estructura de un IDS

Dentro de los sistemas de detección de intrusiones existe una gran diversidad de formatos y arquitecturas. Desde hace tiempo se está intentado unificar este tipo de tecnologías, dando lugar a los denominados Common Intrusión Detection Framework (CIDF), tal y como muestra la Ilustración 3. Los sistemas de detección de intrusiones (IDS) generan eventos y se comunican entre los diferentes componentes del sistema mediante mensajes. Los generadores de eventos, suelen ser agentes o sondas que tienen como objetivo la obtención de datos del exterior hacia sistema de detección de intrusos. La entrada de información suele ser en un formato bruto. Sobre el mensaje en bruto se realiza un pre-procesamiento de la información, con el fin de obtener eventos comprensibles por los demás componentes. Estos componentes se conocen como **E-boxes.**

Los IDS poseen un motor de inferencia capaz de evaluar la relevancia de los eventos recibidos de los generadores de eventos, y generar como salida nuevos eventos. Este tipo de motor se denomina "motor de análisis", y se considera el componente más crítico. Estos motores, se basan en reconocedores de patrones, sistemas estadísticos, sistemas de correlación de eventos, entre otros. Sobre este componente recae la responsabilidad de analizar el flujo de eventos y de extraer la información relevante. Se conocen como componentes **A-boxes**.

Existe un componente que es el encargado de almacenar y albergar físicamente el conocimiento del motor de análisis, conteniendo los eventos generados. Este componente se denomina **D-box** o unidad de almacenamiento. Una vez, que los eventos han sido recogidos, analizados y almacenados, es necesario emitir una respuesta. El componente encargado de realizar esa acción, es el responsable de reaccionar basándose en los mensajes enviados por los demás componentes del IDS. Por otra parte, los **R-boxes** o unidades de respuesta están orientados a prevenir los ataques o a cortar un ataque que se está realizando en ese momento. La reacción a estos ataques puede verse desde dos puntos de vista, uno sería una actitud pasiva frente a un ataque, consistiendo simplemente en el aviso o emisión de una alerta del sistema de detección, que se está produciendo una acción maliciosa, o una actitud activa, donde el sistema toma la decisión de actuar y cortar la acción maliciosa **[VER 04]**.

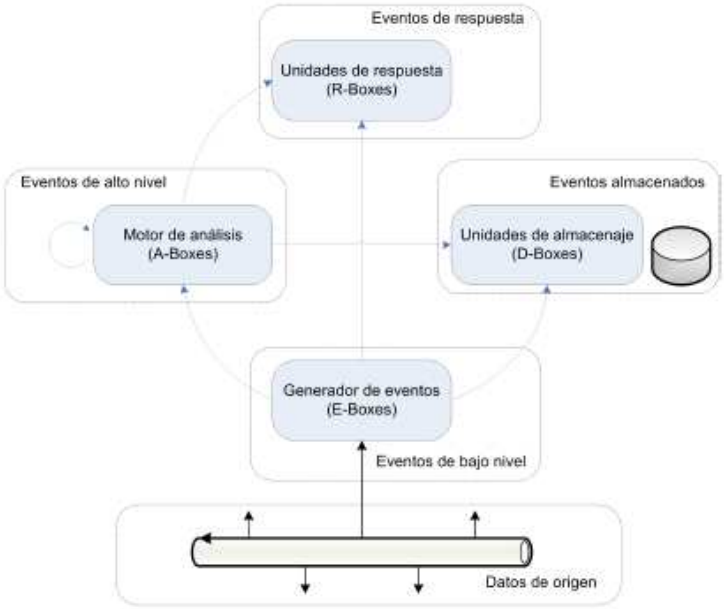


Ilustración 3 Arquitectura CIDF

2.2.3 Clasificación de Sistemas de Detección de Intrusiones

A continuación se expone diversos tipos de sistemas de detección de intrusiones (IDS) según el origen de los datos:

Los **Host Intrusion Detection System** o **HIDS,** son sistemas de detección de intrusiones "host", su objetivo es detectar anomalías que indiquen un riesgo potencial monitorizando las actividades en una o varias máquinas. Los Host IDS, se instalan en una sola máquina, aunque pueden también ser instaladas sobre máquinas que comprometen la red, como servidores o estaciones de trabajo. Están diseñados para monitorizar, detectar y responder a los datos emitidos desde cualquier fuente. Los sensores, instalados dentro de las máquinas, generan datos que son recogidos y validados por el HIDS.

La información que recolectan los HIDS suelen ser registros del sistema operativo o de determinadas aplicaciones, el estado de la CPU, el estado de la memoria, toda esa información es recogida en forma de logs, pudiendo analizar las actividades de la máquina comprometida con una gran precisión y lo que es más importante, contextualizarlo. Por otra parte, los sistemas que poseen este tipo de mecanismo, deben proteger adecuadamente la integridad de los logs generados, ya que este sistema de detección requiere la confianza de dichos registros. A modo de ejemplo, Tripwire es una aplicación de OpenSource o código abierto que monitoriza y alerta de los cambios específicos producidos en determinados ficheros.

Los **Network Intrusion Detection System** o **NIDS** son sistemas de detección de intrusos por red. Los NIDS consisten en un conjunto de máquinas y agentes colocados en puntos críticos de la red que se desea proteger. Los agentes monitorizan el tráfico, realizando un análisis de la información local y transmitiéndoselo a un gestor central. Este tipo de sistema detecta un ataque por medio de la captura y el análisis de los paquetes que viajan por la red o por los dispositivos que están en ella, gracias a que disponen de una o varias interfaces conectados a puntos estratégicos. El NIDS realiza la monitorización del tráfico que pasa por la red, protegiendo así a todas las máquinas que forman parte de ella, en busca de ataques maliciosos.

Estos sistemas suelen ser pasivos y generalmente se colocan en routers y cortafuegos. No interfieren con el uso normal y correcto de la red y permiten una detección de ataques con un mayor nivel de abstracción, debido a la multitud de máquinas que monitorizan. Funcionan como sniffers de red.

La ubicación de este tipo de sistemas suelen encontrarse delante o detrás de un cortafuego o en combinación de ambos casos. Si situamos un NIDS delante de un cortafuego, se pueden monitorizar los ataques efectuados a nuestra red. En el caso de colocar un HIDS detrás del cortafuegos, analizará el tráfico que no haya podido filtrar el cortafuegos, por lo que se obtendrá un número menor de logs y se sabrá con exactitud que ataques han sido los más efectivos. Es posible asociar determinados ataques que ocurren a un lado y al otro del cortafuego. Si se decide optar por este sistema, es necesario implementarlo en dos máquinas. También existe la posibilidad de tener una máquina que realice las funciones de cortafuegos y NIDS, dando como resultado la primera opción comentada anteriormente.

Debemos tener en cuenta, que la máquina que contenga el NIDS debe tener un trato de seguridad especial., No serviría de nada almacenar todas las amenazas, si el atacante es capaz de borrar los logs de registros de las mismas. Podemos concluir que estos sistemas pueden monitorizar el tráfico externo e interno, además son bastante sencillos y rápidos de instalar.

Snort es una aplicación comercial de tipo Open Source o código abierto. Es considerado un sniffer o escuchador muy flexible, capaz de almacenar sus registros tanto en ficheros de texto como en una base de datos asociada.

Los **Hybrid Intrusion Detection System** o **Hybrid-IDS,** son sistemas de detección de intrusiones híbridos. Es una combinación de los dos sistemas anteriormente mencionados y realizan una monitorización del sistema local, además del tráfico de red. Normalmente un Hyrbrid-IDS está formado por sensores situados en cada máquina de forma local y en cada segmento de red que se considere oportuno. Este sistema permite obtener una gran cantidad de información, facilitando la detección de un ataque antes de que pueda ser lanzado en la red administrada. Una aplicación Open Source que utiliza este sistema es Prelude. Posee una arquitectura

distribuida con canales autenticados y encriptados, además de una gran cantidad de sensores para diferentes sistemas operativos.

Los **Intrusion Prevention System** o **IPS,** son sistemas de prevención de intrusiones, pueden combinar las capacidades de bloqueo de los cortafuegos con el análisis profundo de la red. Se puede considerar un IPS, cuando cualquier dispositivo pueda realizar la acción de detectar y prevenir ataques. La diferencia de este tipo de sistemas respecto a los demás sistemas de detección de intrusiones, es que permite definir políticas de seguridad, que conllevan a la ejecución de determinadas acciones con el fin de proteger el sistema comprometido. Se considera que un IPS es un sistema que protege de forma activa, adoptando medidas correctivas inmediatas, en cambio los anteriores IDS lo protegen de forma pasiva. La aplicación Snort, puede ser configurado como un sistema de prevención de intrusiones **[SAL 05].**

2.2.4 Limitaciones de los IDS

Los sistemas de detección de intrusiones han sido de gran utilidad en el campo de la seguridad informática, aunque estas herramientas han sido y siguen siendo muy útiles, posee ciertas carencias y limitaciones.

La acción de detectar una intrusión, es una maniobra bastante compleja. Los ataques suelen ser de naturaleza muy diversa, es muy difícil predecir totalmente su comportamiento. Es imposible diseñar un sistema de protección que garantice la totalidad del sistema. Aunque ciertos ataques suelen converger hacia patrones similares, es necesario realizar continuas actualizaciones y estudios de los mismos.

Los sistemas que hacen uso de IDS suelen padecer una disminución del rendimiento en condiciones normales. Un IDS requiere un mantenimiento diario. Además en algunas ocasiones suele requerir la supervisión por parte de una persona técnica, que esté pendiente de las posibles alertas que se notifiquen. Dentro de este mantenimiento, formaría parte el ajuste de los filtros, para obtener los mínimos falsos positivos posibles.

Generalmente los IDS suelen ser pasivos. Si durante un atacante se consigue desbordar o consumir todos sus recursos del sistema comprometido, puede llevar a que se alcance su inutilidad. Si esta situación ocurriese, se podría atacar libremente el sistema. Asimismo, existen mecanismos en el cual, si un sistema se desborda y queda desactivado, este como medida de seguridad, cierra todas las conexiones de red que esté protegiendo en ese momento.

La evasión e inserción, son dos tipos de problemas de los IDS. Un claro ejemplo es el caso de los NIDS, estos sólo tienen conocimiento de la información de los segmentos de red, pero no de la topología de la red, ni de las máquinas que la forman. La *evasión* consiste en que un sistema final acepta un paquete que el sistema

de detección rechaza. De esta forma se perdería la información transmitida. En el caso de la *inserción* el sistema de detección acepta un paquete, que el sistema final rechazaría. El sistema de detección permite el acceso de un paquete que cree que el sistema final aceptaría y no es así. Un atacante puede explotar este tipo de condiciones. Esta situación es provocada por un desajuste entre el sistema de detección y el sistema final.

Otro problema es el de la identificación de falsos positivos. Se considera que un IDS está comprometido, cuando identifica como intrusión una acción que no era. Este problema puede disminuir a través de procesos de fine-tunning.

Por último hay que señalar que cuando se recibe un evento, dicho evento se analiza y se compara contra toda la base de reglas definidas. Si alguna de estas reglas coincide con la asociación de los eventos recibidos se dispararán las alarmas y se genera una notificación. La modificación del contenido de los paquetes o parámetros de un determinado evento, puede resultar que el ataque pase desapercibido para el motor de correlación, no ejecutándose así ninguna regla. Esta forma de asociar la información en forma de eventos, a través de reglas predefinidas, se considera comúnmente como proceso de correlación **[BAR 05]**.

2.3 Logs o registros

Un Log es un registro oficial de una serie de eventos que se obtienen durante un periodo de tiempo determinado. En el ámbito de la seguridad informática, los logs son usados para registrar información sobre qué, quién, cuando, donde, por qué. En ingles responden a las preguntas de “what, who, when, where y why”, denominadas las 5W, sobre eventos que se suceden sobre un dispositivo o una aplicación en particular.

La gran mayoría de los registros son almacenados en un formato estándar ASCII, que son un conjunto de caracteres comunes para dispositivos y aplicaciones, por lo que existe la compatibilidad entre dispositivos para ser leídos, empaquetarlos y desplegarlos **[WEB04]**. Del análisis de los logs, se puede determinar la acción necesaria para corregir un problema o realizar un análisis más profundo en un caso claro de violación de la seguridad establecida.

Existe un problema común con la veracidad y evidencia de los logs. Se puede hablar de evidencia común cuando la información utilizada para decidir si las proposiciones utilizadas en una disputa son ciertas. La veracidad de la información es asegurada o comprobada mediante una prueba de confiabilidad de la fuente. La evidencia común debe respetar los conceptos de autenticidad, exactitud y suficiencia de la información registrada.

Los registros de logs pueden consumir una gran cantidad de almacenamiento, en el sistema central que los almacena. Existe la dificultad de no saber si la

información almacenada será útil en una futura investigación legal. Una forma de estructurar y organizar la información de los registros consiste en el concepto de rotación de los mismos. La rotación de los logs permite limitar el volumen de datos almacenadas para que su análisis se más manejable. La política de rotación de logs consiste, en sustituir el fichero de log actual utilizado una vez alcanzado su límite, por otro fichero de log nuevo. El tamaño de un log o el intervalo de tiempo de rotación, puede establecerse en horas, días, semanas o meses, según lo crítico de la información. Una vez decidida la política de rotación, se produce un renombramiento del nombre del registro y una posible nueva ubicación de este. Cuando una rotación se ha realizado de forma exitosa, se habrá generado un log rotado, con un nombre o identificación asociado a un número, junto a un nuevo fichero de log, con el nombre natural de ese log.

Es importante que todas aquellas fuentes generadoras de logs, estén sincronizadas en tiempo de forma exacta. Además de su influencia en el proceso de correlación, en el momento en el que se produzca un incidente se podrá buscar la información de una manera eficiente, ya que estos podrán ser encontrados de forma centralizada, organizada y exactos en el tiempo **[WEB05]**.

2.3.1 Administración y almacenamiento de logs

En el momento del almacenaje y administración de registros, existen unos factores a tener en cuenta:

- **Volumen de logs**, se tendrá que decidir si se opta por un tratamiento distribuido, siendo este más complejo y escalable o una opción centralizada, sencilla y más limitada.

- La **heterogeneidad de formato de logs**, es un problema a la hora de la correlación de los mismos. El establecer un formato común de estos, una vez almacenado el registro original, se hace casi imprescindible.

- La **arquitectura de las redes y de los sistemas**, junto con el ancho de banda, las capacidades de almacenamiento, la infraestructura y políticas de seguridad, juegan un papel relevante.

El análisis de logs se ha venido realizando generalmente de forma reactiva, es decir, ante indicios de un incidente o por requerimiento externo. Actualmente se realiza de forma proactiva, identificando la actividad sospechosa a medida que se produce, casi en tiempo real, facilitando la toma de decisiones de acciones correctivas apropiadas a la situación identificada **[SHE 09]**.

2.3.2 Syslog

Syslog fue desarrollado por Eric Allman como parte del proyecto Sendmail en los años 1980. Se comprobó que esta herramienta era bastante útil y sencilla, por lo que otras aplicaciones también lo usaron. Syslog está presente en sistemas Unix y GNU/Linux y en diferentes implementaciones de sistemas operativos como Microsoft Windows. Syslog es un sistema de eventos que se encarga de la administración de logs generados por eventos del sistema, por el núcleo del sistema operativo o por alguno de sus programas. Syslog utiliza una aplicación ejecutada de forma silenciosa o en segundo plano, que se encarga de capturar cualquier log generado. A esta aplicación que se ejecuta en segundo plano se llama syslogd. El protocolo Syslog, los mensajes de formato Syslog se suelen enviar por el puerto 514 vía UDP en texto plano. Algunas implementaciones como syslog-ng permiten usar TCP, ofreciendo conexiones cifradas seguras para clientes o servidores que no utilizan TLS o SSL de forma nativa. Actualmente syslog en un estándar recogido en los "Request For Comment" o RFC, con continuas extensiones y mejoras de seguridad **[WEB06].**

2.4 Eventos

Un evento es considerado cualquier acontecimiento, circunstancia, suceso o caso posible. Este acontecimiento puede ocurrir en una circunstancia determinada y es generalmente un hecho imprevisto o un cambio significativo de estado. Los eventos pueden ser de diferentes tipos, tal y como se describe a continuación.

Los *eventos derivados*, consisten en que un evento se genera como resultado de la aplicación de un método o proceso, en función de uno o más eventos. Un ejemplo de evento derivado es la ausencia de un evento en un intervalo de tiempo dado. La ausencia del primer evento provoca un evento derivado que informa que el primer evento esperado no ha ocurrido. Este ejemplo representa un evento de tipo temporizado.

Un *evento complejo*, es un evento que representa una abstracción de otros eventos denominados miembros. Los eventos complejos no implican que lleven consigo siempre un conjunto de eventos miembros. Mientras que los eventos derivados pueden estar formados por un solo acontecimiento, los eventos complejos pueden formarse por agrupación de múltiples acontecimientos.

Los *eventos compuestos*, consisten en la recogida de los acontecimientos que cumplen un cierto patrón. Un evento compuesto representa una abstracción, y eventos derivados, como resultado de un proceso de cálculo.

En la ilustración 4, podemos observar la estructuración y composición de los tipos de eventos, anteriormente definidos:

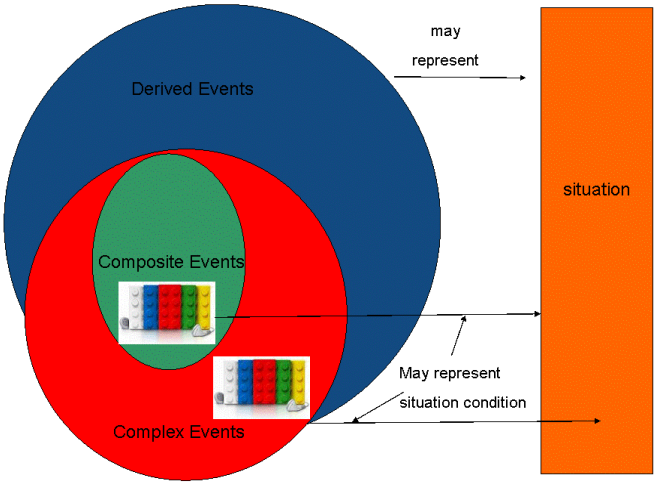


Ilustración 4. Relaciones entre tipos de eventos

2.5 Modelo de procesamiento de eventos

El modelo de procesamiento de eventos ayuda a los sistemas que necesitan cambiar y responder, a las rápidas y cambiantes condiciones de su negocio. En este apartado, se comenta, las diferentes técnicas utilizadas en la actualidad, que hacen posible este modelo.

2.5.1 Event Driven Architecture o EDA

EDA representa un patrón arquitectónico basado en la promoción o publicación, detección, consumo y reacción de los eventos. Este patrón puede ser aplicado en el diseño e implementación de aplicaciones que trasmiten los acontecimientos entre sus componentes software, siendo estos componentes prácticamente independientes al sistema **[WEB07]**.

Un evento que es generado por un sistema, y es emitido desde un agente de eventos asociado a dicho sistema, estos eventos son dirigidos hacia los componentes consumidores de la arquitectura. Los componentes que forman la arquitectura basada en eventos, tienen la responsabilidad de aplicar una reacción tan pronto como se presente un evento. La reacción puede ser o no aceptado totalmente por el primer componente que recibe el evento, o enviada a un componente superior, que puede tener la responsabilidad de filtrar, transformar y remitir en otro componente, el evento recibido inicial.

Los sistemas que implementan la arquitectura EDA, permiten que las aplicaciones que lo implementan, faciliten su capacidad de respuesta, en ambientes

impredecibles o asíncronos. Existen tres estilos generales de procesamiento de eventos, utilizadas en este tipo de arquitecturas:

1. El procesamiento de *eventos simples* se refiere a un caso sencillo de tratamiento de acontecimientos que están directamente relacionados con cambios específicos de una determinada condición.

2. El procesamiento *de flujos de eventos (ESP)* engloba tanto a eventos ordinarios como acontecimientos notables. Este tipo de procesamiento es comúnmente utilizado para manejar en tiempo real el flujo de información entorno a un determinado contexto, permitiendo la toma de decisiones en ese tiempo.

3. El procesamiento de *eventos complejos (CEP)* permite a los eventos simples y ordinarios combinarse de forma que se considere un evento complejo. El procesamiento de eventos complejos evalúa una confluencia de acontecimientos y su posterior toma de decisiones, Los acontecimientos pueden producirse en periodos cortos y largos de tiempo. La correlación de eventos, puede ser causal, temporal o espacial. CEP exige el empleo de sofisticados intérpretes de eventos, definiciones de patrones y concordancia junto a las técnicas de correlación.

Podemos observar en la ilustración 5, los componentes básicos de una arquitectura orientada a eventos, como componentes principales destacar el colector (collector) de eventos y el motor de correlación (correlation engine). El primero se encarga de recibir los eventos que se desean analizar y el segundo tiene la responsabilidad de procesar los eventos, generando o no un respuesta de forma inmediata **[WEB08]**.

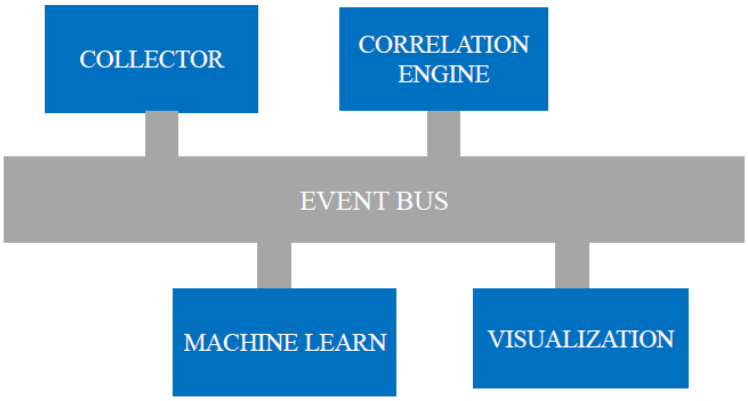


Ilustración 5 Componentes básicos en arquitecturas orientadas a eventos

2.5.2 Event Stream Processing o ESP

Son un conjunto de tecnologías diseñadas para ayudar en la construcción de sistemas orientados a eventos (EDA), basándose en flujos de eventos **[WEB09]**. Las tecnologías ESP incluyen la visualización de eventos, lenguajes de procesamiento de eventos (EPL) o procesamiento de eventos complejos (CEP). La tarea de procesamiento consiste en la identificación de eventos significativos entre múltiples flujos de eventos que representan datos, empleando técnicas, como la detección de patrones complejos, correlación de eventos, entre otros. Las aplicaciones prácticas de ESP son múltiples, a través de algoritmos comerciales para servicios financieros, detección de fraudes, monitorización de procesos, en general aplicaciones que requieran una respuesta en tiempo real **[WEB10]**.

2.5.3 Complex Event Processing o CEP

El Procesamiento de Eventos Complejos, es un conjunto de herramientas que nos ayudan a entender, controlar, y procesar eventos generados de carácter complejo. El acrónimo "CEP" también se utiliza para identificar a una clase de software que utilizan aplicaciones basados en eventos **[WEB11]**. CEP considera evento a cualquier acontecimiento que ocurra o que se contemple como sucede. Un evento puede ser un objeto que representa, codifica o registra un suceso informático. CEP, se basa en el enlace existente entre los patrones de eventos que se procesan. Este procesamiento es el resultado de aplicar una serie de disciplinas científicas que van desde algoritmos de dominio de lenguajes específicos para modelos semánticos, a métodos matemáticos de comprensión, detección de patrones y teorías probabilísticas.

Una idea fundamental en CEP, son las relaciones de sucesos o causalidad. Los patrones de acontecimientos, se define como una combinación booleana de eventos como A y B, que definen un patrón. Este concepto se aproxima bastante al mecanismo de consulta sobre una base de datos. Estas relaciones tiempo o causalidad entre eventos, pueden ser entendidas como una combinación de variables booleanas **[WEB12]**.

El objetivo de CEP, es descubrir la información útil contenida en los acontecimientos que se suceden a todos los niveles de la organización y analizar su impacto como caso complejo y luego ejecutar un plan de acción en tiempo real. El volumen de datos con los que se trabaja es bastante grande, por lo que el uso de una base de datos es poco eficiente en este caso, debido a los tiempos de carga que poseen este tipo de herramientas **[AYL 08]**.

En la ilustración 6, se intenta ejemplarizar el proceso de gestión de eventos a través de componentes ESP/CEP:

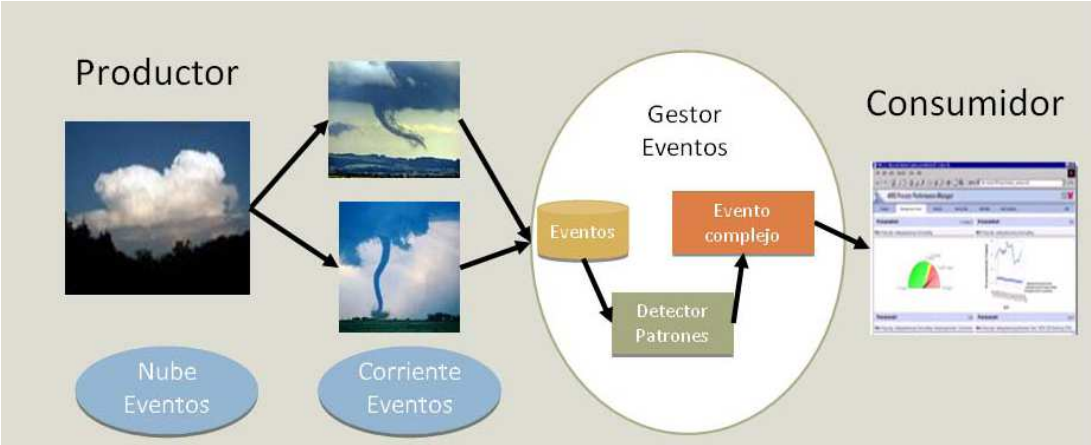


Ilustración 6 Componentes de ESP/CEP

En esta ilustración podemos observar como el productor introduce un nuevo evento dentro de la nube de eventos, donde son gestionados gracias al componente de gestor de eventos. Estos eventos que provienen de la corriente de eventos, son procesados, aplicando mecanismo de detección de patrones con el objetivo de generar eventos complejos. Los eventos complejos son recibidos por el consumidor que interpretará la información facilitada según como este configurado.

Si se recibieran eventos que representaran un número de intentos de conexión a un puerto determinado de forma anormal, esta situación representa una amenaza, para el mantenimiento del servicio que el sistema esté ofreciendo. Como ejemplo de escenario real, daría lugar a que el motor de correlación recibe más de cien intentos de conexión hacia un puerto determinado, en menos de 2 segundos. Esta acción está predefinida en una regla de correlación como un acontecimiento que resulta una amenaza, por lo que se lanza el evento complejo “ataque a puerto activo”.

Una vez que el evento de “ataque a puerto activo” es procesado, se tomará las acciones oportunas para evitarlo. Una posibilidad de respuesta proactiva, es la acción de cerrar el puerto atacado del sistema amenazado, otra posibilidad es emitir un aviso al administrador del sistema, para este evalúe el plan de actuación.

Otra circunstancia es, si el sistema central de eventos no recibiera eventos de la máquina monitorizada durante un largo periodo de tiempo, provocaría una situación de amenaza. Como ejemplo de escenario real, si no se recibe cualquier tipo de evento del sistema crítico monitorizado durante 15 minutos, se lanzará un evento de “conexión interrumpida”. La aparición del evento “ataque a puerto activo”, junto con el evento “conexión interrumpida” en un periodo de tiempo corto, dará lugar a un evento de nivel superior denominado “ataque por denegación de servicio”. Este evento es considerado un evento complejo compuesto de un nivel superior, ya que está formado por la combinación de otros eventos de menor nivel en la jerarquía de eventos.

2.5.4 Event Processing Language o EPL

Es un lenguaje de programación diseñado específicamente para el procesamiento de eventos. EPL cuyo significado traducido al castellano es lenguaje procesamiento de eventos, extiende y amplia lenguajes tipo SQL o Structured Query Language, basados en consultadas y operaciones sobre bases de datos. EPL es capaz de definir reglas temporales, además de aplicar reglas no lineales para el procesamiento de eventos. Su sintaxis es muy similar a los lenguajes SQL. En la actualidad se trabaja con lenguajes EPL muy similares a SQL **[LUC 08].**

2.6 Correlación

La correlación de eventos es una parte importante de la gestión de eventos de seguridad. Su automatización junto con la integración con el resto de procesos de gestión puede facilitar enormemente las operaciones de supervisión de forma eficiente. Por correlación se entiende una relación entre dos ó más entidades que son comparadas en un periodo de tiempo determinado.

La correlación de eventos apareció inicialmente unida a las herramientas de gestión y monitorización de redes. Estas herramientas se utilizaban para evitar las denominadas “Nubes de Eventos” o “Event Storms”, que representa una gran cantidad de eventos generados tras una caída de un dispositivo determinado. Desde el punto de vista de la seguridad, la correlación se entiende como la agrupación de múltiples elementos individuales para la determinación de situaciones anómalas o ataques en los diferentes componentes de la arquitectura a proteger.

Los sistemas de gestión o detección de intrusiones, necesitan utilizar la correlación por varios motivos, como por ejemplo:

- Enormes cantidades de datos y registros difíciles de procesar, que limitan la investigación de incidentes.

- Integración y unificación de múltiples plataformas y sistemas de diversos fabricantes con formatos y registros diferentes.

- Gestión de falsos positivos producidos por el trafico de Internet, ataques automatizados, que deben ser notificados por los sistemas de detección de intrusiones.

- La pérdida de tiempo y recursos, como consecuencia del análisis de grandes volúmenes de datos.

- Carencia de metodologías de revisión y análisis de logs o registros del sistema.

La información que puede manejar un motor de correlación, puede obtenerse a través de registros de seguridad o auditorias, como son los registros generados por cortafuegos, antivirus, sistemas de detección de intrusiones, herramientas de búsquedas de vulnerabilidades, herramientas de filtrados de contenidos, sistemas manuales de análisis de seguridad, entre otros. El ámbito de uso de los sistemas de correlación, es muy amplio. Puede utilizarse principalmente en entornos críticos, como son las entidades estatales con una infraestructura controlada a través de sistemas de información. Cuanto más grande y heterogénea sea la organización, más útil es el uso de un sistema de correlación. Existen múltiples beneficios en el uso de sistemas de correlación, entre los que podemos mencionar:

- Mejora de la seguridad global de organización, con la posibilidad de analizar los incidentes y sus posibles causas.

- Obtener una visión completa de la seguridad de la organización desde un único punto y en tiempo real.

- Centralizar eventos de seguridad y facilitar su tratamiento y acceso.

- Facilita la capacidad de responder ante un incidente, con toda la información necesaria.

- Disminuye los costes económicos en contratación de empleados técnicos que realicen esa misma función, e incrementa la eficiencia.

- Posibilidad de obtener una infraestructura distribuida y escalable **[OSO 06]**.

2.6.1 Tipos de correlación

Podemos diferenciar la correlación según el tipo de datos a comparar.

Micro Correlación: Compara fuentes de datos del mismo tipo generados por diferentes fuentes, como sería la búsqueda de todos los eventos que se generan bajo una misma IP.

- **Correlación atómica:** Se realiza sobre el mismo tipo de dato no normalizado o no transformado a un formato común.

- **Correlación de campos**: Se relacionan eventos dados un único campo o múltiples campos del evento, dentro de los datos normalizados, como son todos los eventos generados sobre una misma IP y destinados bajo un mismo puerto.

- **Correlación mediante reglas o patrones**: Correlación de un conjunto de eventos relacionados a través de reglas y patrones preestablecidos con anterioridad.

- **Correlación de firmas**: Es un tipo de correlación que se utiliza en los sistemas de detección de intrusiones, donde se comparan datos sospechosos con patrones predefinidos como maliciosos.

Macro Correlación: Compara múltiples tipos de datos de diferentes sistemas, como sería la comparación de un evento generado por un sistema de detección de intrusiones y un informe de una aplicación que analice vulnerabilidades.

- **Correlación de perfiles**: Compara eventos sucedidos en un momento determinado con los eventos originados por ataques en el pasado.

- **Correlación estadística**: Correlan métricas actuales con las equivalencias históricas en diferentes periodos de tiempo, como sería la comparación de intentos de acceso fallidos en un entorno en un mes.

- **Correlación mediante listas**: Comparación de eventos normalizados con una serie de listas predefinidas de ataques anteriores. Las listas se actualizan automáticamente con cada nuevo incidente de seguridad, sería el análisis de las direcciones IP de los eventos con una lista de ataques conocidos.

- **Correlación dirigida a eventos**: Utiliza la información de cualquier fuente como medio para ayudar en el diagnostico, resolución y la prioridad del evento. Posibilitando el uso de un inventario para asignar prioridades a los eventos según la importancia del entorno que afecten.

Correlación múltiple: Son aquellos tipos cuya correlación intervienen fuentes de datos generados por múltiples organizaciones o instituciones **[RUI 03]**.

2.6.2 Motor de correlación Esper

Entre los diferentes motores de correlación comerciales que existen actualmente, se ha elegido el motor de correlación Esper escrito en Java, por ser una implementación Open Source del concepto CEP/ESP y que vamos a utilizar en este trabajo. Esper permite un desarrollo rápido en aplicaciones que necesitan el procesamiento de grandes cantidades de mensajes entrantes o eventos y responde bajo determinadas condiciones en tiempo real. El motor de Esper ha sido desarrollado para satisfacer los requisitos de las aplicaciones que necesitan analizar y reaccionar ante los acontecimientos.

Desde un punto de vista CEP/ESP, el flujo de datos en tiempo real junto la posibilidad de realizar consultas, garantiza la reactividad inmediata y reduce la carga de desarrollo. Esper implementa el concepto de procesamiento de flujos de eventos (ESP) y un motor de correlación de eventos (CEP). Además resuelve un complejo conjunto de problemas que en la actualidad se presentan en las arquitecturas orientadas a eventos.

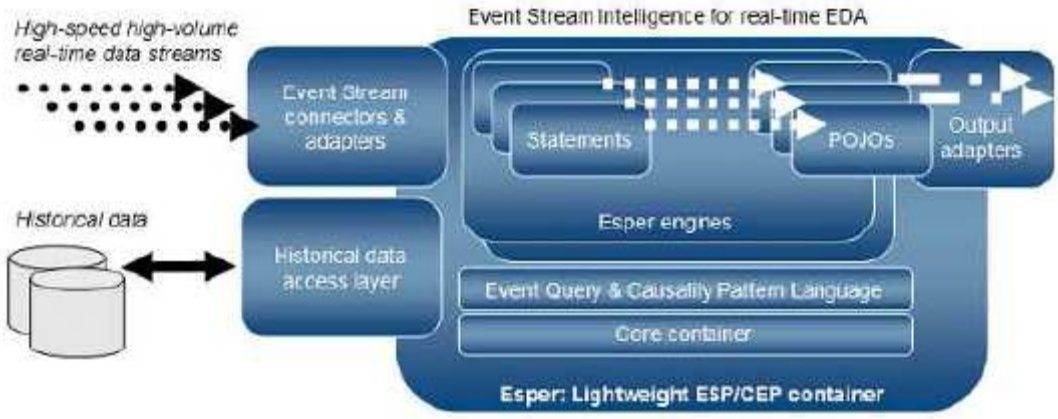


Ilustración 7 Arquitectura básica del motor Esper

La ilustración 7 muestra la arquitectura básica del motor Esper. Sus componentes principales son los siguientes:

- **Statements**: Son las sentencias de correlación del motor de Esper.
- **POJOs**: Son objetos que utiliza el motor Esper, en representación de los diferentes tipos de eventos de forma interna.
- **EventQuery & Causally Pattern Language:** Representan un lenguaje similar a SQL para la detección de patrones complejos de correlación.
- **Core container:** Ofrece una interfaz de comportamiento común a múltiples motores y gestores de eventos.

Esper está diseñado para la correlación de un alto volumen de eventos, donde miles de eventos, pueden hacer imposible su almacenamiento y consulta.

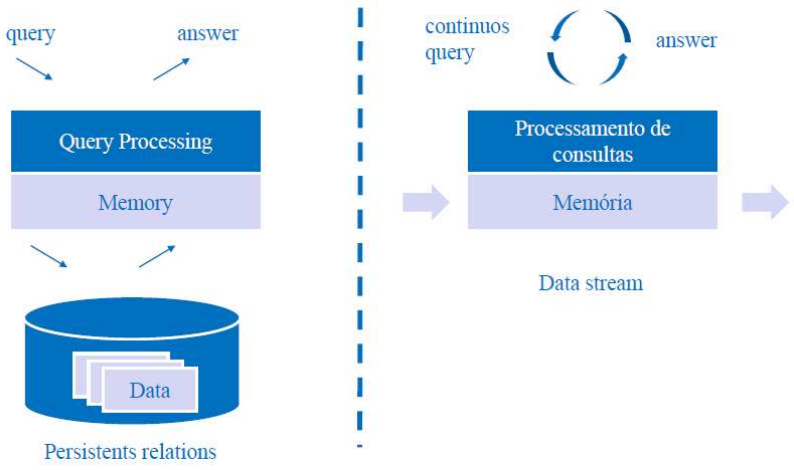


Ilustración 8 Cambio significativo en el procesamiento de flujo de eventos

En la ilustración 8, podemos observar el cambio significativo con respecto al proceso de consulta clásico de una base, en comparación, con el de un motor de correlación. Las consultas en el modelo clásico de una base de datos, se realizan en dos partes. La primera, se produce el procesamiento de la consulta en memoria y en segundo lugar, se accede a disco, para obtener la respuesta. En un motor de correlación, el procesamiento de consulta es continuo. Las consultas son procesadas en memoria, por lo que permite una mayor rapidez de respuesta, con respecto al acceso a la base de datos. También comentar, que un motor de correlación, requiere un mayor uso de recursos de memoria dentro del sistema.

Esper proporciona un lenguaje de procesador de eventos (EPL) para poder utilizar herramientas del tipo de filtros, agregaciones, ventanas de tiempo sobre flujos de múltiples eventos. También proporciona una semántica propia para expresar la causalidad temporal entre eventos complejos. Permite agregar eventos en tiempo real, evitando problemas en el momento de la implementación, como son la serialización de los mensajes y la alta latencia en la red por cada mensaje recibido.

La arquitectura de Esper se basa en un contenedor ligero formado por múltiples componentes, como son los diversos motores de gestión de eventos, la representación interna de los eventos procesados (POJO), la detección de patrones complejos a través de su EPL y las diferentes interfaces Java útiles en la notificación de eventos. Todos estos componentes permiten un excelente rendimiento del flujo de eventos y la ejecución de complejas consultas de forma continua que correlacionan distintas corrientes de eventos para detectar un patrón buscado.

Este tipo de motores son muy útiles en los casos donde la toma de decisiones en tiempo real, sobre la detección de un conjunto de acontecimientos en un periodo corto de tiempo es realmente crítica **[WEB13]**.

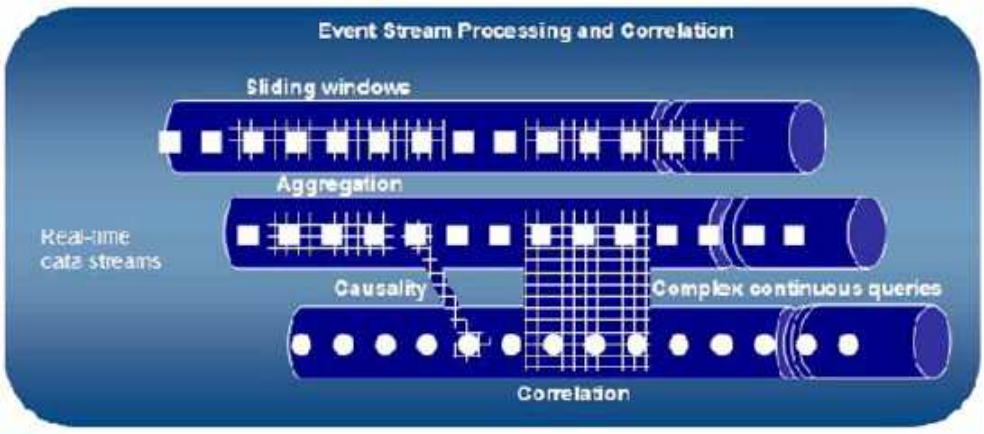


Ilustración 9 Modelo de procesamiento y correlación de Esper

En la ilustración 9, observamos el modelo de procesamiento de flujo de eventos y la correlación de los mismos en el motor de Esper. Los eventos se agregan al motor de correlación a medida que se reciben. Además, la existencia de un periodo de tiempo acotado en la recepción de eventos (ventana de tiempo), ayuda a ordenarlos y da lugar, a relaciones de causalidad con los demás eventos que forman parte de la nube de eventos.

Capítulo 3
Descripción informática

En este capítulo vamos a describir el problema que se pretende resolver y la solución propuesta[1]

3.1 Descripción del problema

Se trata de implementar un sistema capaz de gestionar todos aquellos eventos cuyo formato sea Syslog, almacenarlos y posteriormente correlarlos. La aplicación debe ser capaz de gestionar el acceso de diferentes roles de usuarios, dependiendo de los permisos que posean.

El técnico que desee acceder a la aplicación, debe realizar el proceso de autenticación a través de un identificador de usuario o login y una contraseña. El técnico, dependiendo de los permisos que disponga, podrá realizar una serie operaciones dentro de la aplicación. El administrador de la aplicación o el usuario con permisos de administrador, una vez autentificado como tal, deberá configurar el entorno adecuado para poder ejecutar las tareas de monitorización. La configuración del entorno del sistema, consiste en la gestión y configuración de roles de usuario, cuentas y tareas necesarias para el correcto funcionamiento de la tarea de monitor.

Para que el técnico pueda ejecutar una monitorización correcta sobre una o varias fuentes de eventos, los pasos a seguir por el administrador deben ser, en primer lugar, la creación de una cuenta de protocolo. Esta cuenta de protocolo, contendrá la información necesaria para establecer una conexión, sobre la máquina remota si fuera necesario. Se debe crear una fuente de logs, del tipo de protocolo que se necesitará para obtener los eventos. La aplicación está diseñada para ofrecer servicio de recepción de eventos en formato Syslog, a través de protocolo Syslog y SSH. En segundo lugar, es necesario configurar un recolector de eventos. Este recolector está asociado a una o varias fuentes de eventos. Cada recolector creado, es único dentro de la aplicación. En

[1] El sistema que se implementará, está basado en el diseño e implementación de un modelo de procesos innovador orientado al almacenamiento seguro e inspección de contenidos digitales auditables. **Convocatoria 2/2008, Acción Estratégica de Telecomunicaciones y Sociedad de la información. Proyectos empresariales de innovación en materia de procesos.**

Proyecto en cooperación:

- SignWare SL
- Telefónica España SA
- UPM (Universidad Politécnica de Madrid)
- MICYT (Ministerio de Industria, Turismo y Comercio)

tercer lugar, es necesario micro correlar los eventos y especificar las de reglas de correlación, que son dependientes de las fuentes que se desea monitorizar. El sistema está basado en la micro correlación, a través de la correlación de campos de un mismo tipo de evento, junto con la especificación de reglas o patrones predefinidos, es decir, configurados anteriormente. La correlación de eventos a través de reglas, debe ser conforme al leguaje de especificación de EPL. Cuando una regla de correlación se cumple, el motor de correlación que posee el sistema, lanzará en tiempo real, una alarma en forma de correo electrónico, avisando del incidente ocurrido. Las alarmas, es necesario que las cree y configure el administrador del sistema, previo a la creación de una tarea de correlación.

Para poder recibir los eventos de las máquinas remotas, es necesario crear una tarea de monitorización. La tarea de monitorización, consiste en agrupar aquellos recolectores, que estén asociados a las fuentes de eventos, junto con las tareas de correlación y sus reglas, que generen las alarmas ya configuradas. Las tareas de correo, envían correos electrónicos, gracias a un servidor SMTP. El administrador debe configurar el servidor principal y secundario de SMTP del sistema y asociarlo a las tareas de correo electrónico existentes. El técnico, será el responsable entre otras operaciones, en realizar la ejecución del monitor. Cuando el monitor se ejecute, se recibirán los eventos de las máquinas remotas. Estos eventos se almacenarán en una base de datos, con el objetivo de poder ser visualizados y analizados en el futuro. Los diferentes usuarios de la aplicación, podrán interactuar con el sistema, a través de un navegador Web. El acceso al servidor donde se aloja la aplicación, se realizará a través de un canal seguro vía HTTPS.

3.2 Análisis y Especificación de requisitos

Los requisitos que se han obtenido a partir de la descripción del problema son los siguientes.

Tabla 1 Requisitos funcionales

Número Req.	Nombre del requisito	Descripción
RF001	Autentificación	El sistema será utilizado por diferentes usuarios, entre ellos los administradores de la aplicación. La autenticación será a través de la introducción de un nombre de usuario y una contraseña en el inicio de acceso de la aplicación.
RF002	Gestión de usuarios	Se podrán crear y eliminar usuarios dentro de la aplicación. Las acciones de alta, baja y edición de usuarios lo podrá realizar el grupo de usuarios asociados al rol de administradores de la aplicación. Cada usuario debe estar asociado únicamente a un grupo. Los usuarios podrán ser visualizados y editados, sin la posibilidad de modificar el nombre de usuario.

RF002.1	Características básicas de un usuario	Un usuario estará identificado a través de un nombre de usuario, una contraseña y una identificación o nombre público. Existirá un campo de repetición de la contraseña con el fin de verificar que se ha introducido la contraseña deseada. Todos los usuarios estarán asociados a uno o varios grupos dentro de la aplicación.
RF002.2	Modificación de los datos de usuario	El usuario actual de la aplicación, independientemente de los permisos que posea, podrá editar su contraseña y su nombre.
RF003	Gestión de grupos	Se podrán crear y eliminar grupos de usuarios dentro de la aplicación. Las acciones de alta, baja y edición de grupos lo podrán realizar aquellos usuarios asociados al rol de administradores de la aplicación. Cada grupo estará formado por varios usuarios. Cada grupo podrá ser asociado únicamente a una política definida en la aplicación. Los grupos podrán ser visualizados y editados sin la posibilidad de modificar el nombre de grupo.
RF003.1	Identificación básica de un grupo	Un grupo estará identificado a través de un nombre de grupo, un nombre público a mostrar, y un comentario que dará información sobre el grupo. Además el grupo estará asociado a una serie de usuarios ya existentes en la aplicación y la selección de una única política.
RF004	Gestión de políticas de usuarios	Se podrán crear y eliminar políticas sobre los grupos de usuarios, representado así los roles dentro de la aplicación. Las acciones de alta, baja y edición de políticas lo podrán realizar aquellos usuarios asociados al rol de administradores de la aplicación. Cada política debe estar asociada de una serie de permisos ya definidos dentro de la aplicación. Las políticas podrán ser visualizadas y editadas sin la posibilidad de modificar el nombre de política.
RF004.1	Identificación básica de una política de usuario	Una política estará identificada por un nombre de política, un nombre público a mostrar, una descripción de la política, y una lista de permisos ya predefinidos dentro de la aplicación.
RF005	Gestión de permisos	Existirán tres tipos de permisos: administración, operación y mantenimiento. El permiso de administración podrá crear, borrar, editar, ver, ejecutar y parar cualquier recurso del sistema. El permiso de operación podrá, ver, ejecutar y parar cualquier tarea o recurso del sistema. Los permisos de mantenimiento, únicamente podrán visualizar los recursos del sistema sin modificar su estado.
RF006	Gestión de fuentes de eventos	Las fuentes podrán ser creadas, borradas, visualizadas y editadas por aquellos usuarios cuyo grupo posea los permisos oportunos.
RF006.1	Definición de fuentes de eventos	Una fuente está definida por un identificador de fuente único, un nombre y una descripción, siendo todos estos campos obligatorios. Además se especificará el tipo de formato asociado a la fuente, así como todas aquellas particularidades propias de cada fuente, incluido el uso

			de una cuenta, si el protocolo de comunicación lo necesitara.
	RF006.2	Formato de Evento Syslog	El formato de los eventos estará formado por los campos de facilidad, severidad, fecha, máquina origen, aplicación que generó el evento, pid de la aplicación y mensaje.
RF007		Gestión de tareas del sistema	Una tarea podrá ser creada, borrada, visualizada y editada por aquellos usuarios cuyo grupo posea los permisos oportunos.
RF008		Tarea de recolección	Una tarea de recolección está definido por un identificador único, un nombre de recolector y una descripción, siendo todos estos campos obligatorios. Además cada recolector tendrá asociado una serie de fuentes, así como la configuración específica necesaria para cada recolector en particular.
	RF008.1	Recolección Syslog	Los eventos de syslog se enviarán vía UDP, a través del puerto 1514, en formato de texto plano.
	RF008.2	Recolección SSH	Se recolectará ficheros de eventos remotos, con formato estándar syslog RFC3164, a través de conexiones remotas usando el protocolo SSH.
RF009		Tarea de correlación	Una tarea de correlación está definido por un identificador único, un nombre de correlación y una descripción, siendo todos estos campos obligatorios. Además cada tarea de correlación tendrá asociado una serie de reglas, formadas por un identificador de regla, la regla de correlación, un identificador de contexto de la eventualidad evaluada y una serie de alarmas.
	RF009.1	Alarmas de correlación	Una alarma será una tarea de correo, asociada a una regla, dentro de la tarea de correlación. El cumplimiento de una regla definida hará ejecutar la tarea asociada a ella. El destinatario del resultado de la alarma, será aquella cuenta asociada al administrador del sistema.
RF010		Tarea de monitorización	Una tarea de monitorización está definido por un identificador único, un nombre de monitor y una descripción de la tarea, siendo estos campos obligatorios. La tarea de monitorización asociará una o varias tareas de recolección y correlación disponibles ya configurados. La ejecución de la tarea de monitor, provocará ejecución interna de cada una de las tareas que la conforman.
RF011		Agenda de cuentas del sistema	Las cuentas podrán ser creadas, borradas, visualizadas y editadas por aquellos usuarios cuyo grupo posea los permisos oportunos, a través de una colección o agenda de cuentas.
	RF011.1	Cuenta de protocolos de comunicación	Toda aquella información referente a los protocolos de comunicación asociada a la obtención de los eventos de las diferentes fuentes que necesiten ser agrupadas, deberá obtenerse de una cuenta. Una cuenta de protocolo deberá estar especificada con un identificador único, un nombre y una descripción, siendo estos campos obligatorios.

	RF011.2	Cuenta de servicio	Toda información de configuración de los servicios internos de la aplicación será conjuntada a través de una cuenta de servicio. Una cuenta de servicio deberá estar especificada con un identificador único, un nombre y una descripción, siendo estos campos obligatorios

Tabla 2 Requisitos no funcionales

Número Req.	Nombre del requisito	Descripción
RNF01	Usabilidad	La aplicación poseerá una interfaz gráfica intuitiva y sencilla por los usuarios del sistema.
RNF01.1	Navegabilidad	El acceso y navegabilidad de las operaciones dentro de la aplicación serán claros e intuitivos. El acceso a las funcionalidades del sistema debe de ser claro y concreto, siempre con la posibilidad de volver al estado anterior si se desea.
RNF01.2	Botones, colores e iconos	Los botones de acción deberán ser etiquetados con la acción que realizan y tendrán un color gris. Los colores generales de la aplicación serán de la gama de azules, negros, naranjas o rojos. Los iconos serán representativos, siendo utilizados únicamente de manera explicativa o resumen de una operación realizada.
RNF01.3	Mensajes informativas y frases descriptivas	Cada operación realizada paseará un mensaje informativo de color rojo, que informe del estado de la acción realizada. Las frases explicativas de las diferentes funciones de la aplicación deberán ser simples y concisas.
RNF01.4	Barra de scroll	Se evitarán siempre que sea posible la barra de scroll sobre las diferentes páginas de la aplicación. Si esto fuera inevitable, estos deberán ser verticales y no horizontales.

Tabla 3 Requisitos hardware/software

Número Req.	Nombre del requisito	Descripción
RH01	Servicio de OpenJMS	La administración de los eventos recogidos por las fuentes será organizado en colas de OpenJMS y almacenadas en la base de datos asociada Derby.
RH02	Servidor Web	El servidor web alojará el portal web, así como el servicio de OpenJMS y el motor de correlación. El servidor de aplicaciones utilizado será Apache Tomcat 6.0. El cliente será un navegador web cualquiera.
RH03	Motor de Correlación Esper	El motor de correlación que se utilizar en la aplicación será Esper de EsperTech Inc.
RH04	Seguridad	La aplicación contará mecanismos que faciliten un canal seguro encriptando los mensajes intercambiados entre el cliente y el servidor, durante el uso de la aplicación.
RH04.1	HTTPS	Se usará HTTPS como canal seguro, para las comunicaciones entre los clientes y el servidor.
RH04.2	Certificado digital	La autenticación por parte del servidor de aplicaciones Apache Tomcat 6.0, se realizará a través de certificado digital.

3.3 Análisis

Una vez que se han especificado los requisitos, describimos el caso de uso general de nuestra solución, tal y como muestra la ilustración 10. Los actores del sistema son los siguientes:

- El **administrador del sistema** es el encargado de configurar las diferentes tareas, gestionar usuarios y cuentas que se precisen. El administrador puede crear y borrar cualquier tarea, también podrá ejecutar, parar y visualizar recursos.

- El **técnico** en correlación de eventos de seguridad es el encargado de configurar y ejecutar las monitorizaciones, recolecciones y correlaciones, así como la definición de alertas y reglas, ajustándose a las necesidades del momento.

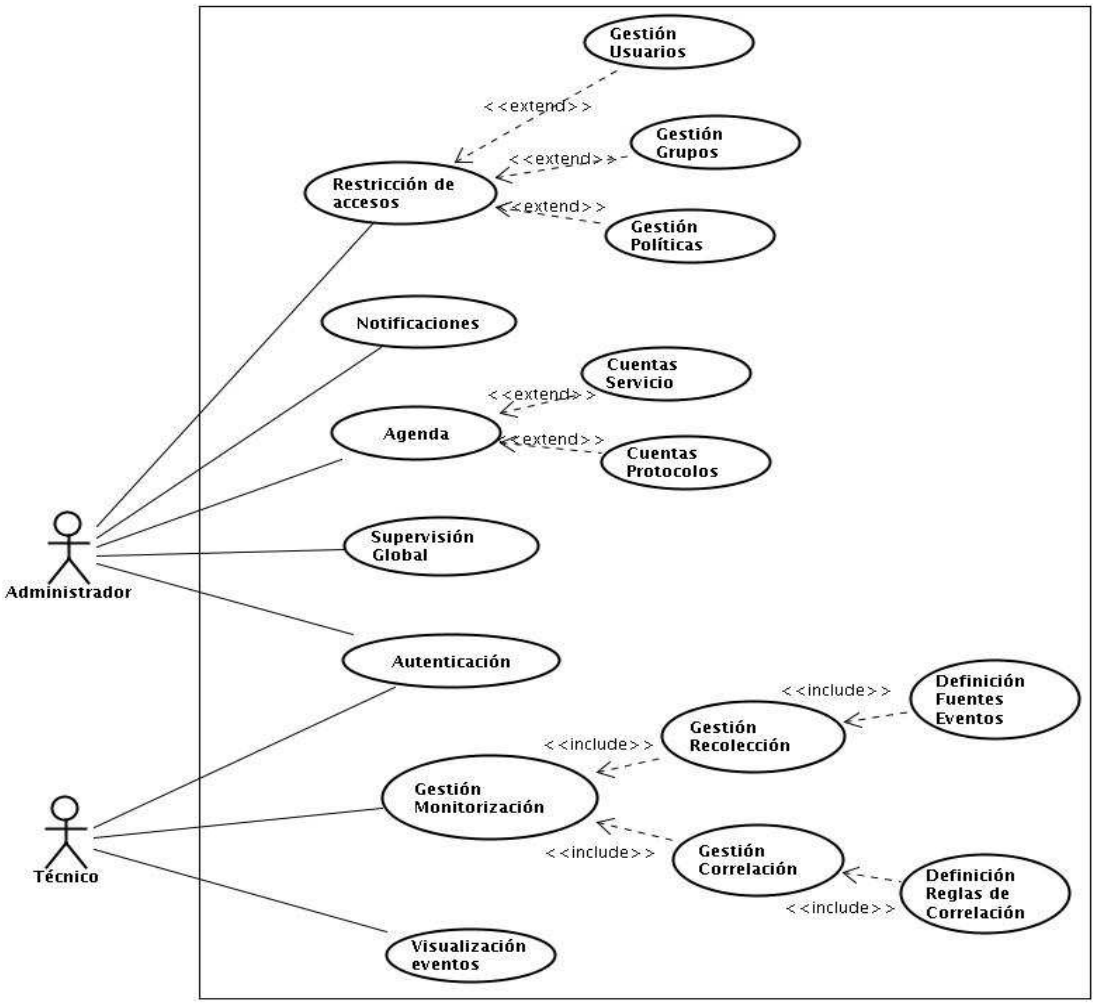


Ilustración 10 Caso de uso general

La descripción de los casos de uso de la ilustración 10 es como sigue:

- **Restricción de acceso al sistema:** Describe como se accede al sistema por parte de los usuarios. El administrador del sistema es el responsable de realizar la gestión de usuarios, grupos y políticas, definiendo los permisos oportunos en cada caso. La operación de gestión incluye la creación, visualización, edición y eliminación, de usuarios, grupos y políticas.

- **Notificaciones:** El administrador envía notificaciones a través de los servicios de correo del sistema, a los demás miembros de la aplicación. Las notificaciones serán informativas sobre eventualidades que puedan suceder a lo largo del mantenimiento de la aplicación.

- **Agenda:** La agenda estará formada por dos usos distintos. Su función es la de organizar los contactos de los técnicos de la aplicación, facilitando la selección de información de interés para el administrador y demás técnicos, como es el caso de las cuentas de correo electrónico, cuentas de la base de datos, entre otras. Desde un punto de vista interno al sistema, se utilizarán cuentas de servicio que ayuden a organizar la información necesaria para el correcto funcionamiento de la aplicación. La gestión de la agenda es responsabilidad del administrador, aunque su uso y consulta lo podrá realizar tanto el administrador como los demás técnicos de la aplicación.

- **Supervisión global:** El administrador podrá supervisar el estado de salud del sistema. Obtiene información en tiempo real de las tareas que se están ejecutando en ese momento, gracias a los logs del propio sistema.

- **Definición de fuentes:** Los técnicos podrán crear, editar, visualizar y borrar fuentes de eventos a través de la definición de estas. Las fuentes representarán los eventos que se gestionarán en la recolección.

- **Gestión de recolección:** Los técnicos asociarán cada una de las fuentes a distintas tareas de recolección. La responsabilidad de los técnicos será la creación, edición, visualización y borrada de cada una de las tareas de recolección especificadas.

- **Gestión de correlación:** La gestión de las tareas de correlación será responsabilidad de los técnicos del sistema. La creación, visualización, edición y borrar de cada una de las tareas, así como la definición de reglas correlativas asociadas a las tareas y la especificación de alertas de las posibles alarmas especificadas, lo gestionarán igualmente los técnicos.

- **Gestión de la monitorización:** La tarea de monitorización agrupará la selección de aquellas tareas de recolección y correlación definidas para la

monitorización de los sistemas críticos. La creación, visualización, edición y eliminación será responsabilidad de los técnicos del sistema.

- **Visualización de eventos:** La totalidad de los eventos que son almacenados en el sistema como resultado de las diferentes monitorizaciones, se visualizarán desde el portal Web. Estos eventos sólo podrán ser visualizados, en ningún momento podrán ser modificados o eliminados.

3.4 Diseño del sistema

En esta fase se mencionan las directrices propuestas durante el análisis con la función de satisfacer los requisitos especificados anteriormente.

3.4.1 Diseño de la interfaz

En diseño del sistema ha primado la necesidad de reunir las propiedades de eficacia y eficiencia en las operaciones realizadas sobre eventos de seguridad, obtenidos de los sistemas críticos monitorizados.

- **Organización de componentes:** Se pretende que la interfaz posea una relación lógica amigable e intuitiva, evitando problemas de confusión e incomodidad operacional. Cabe destacar, su disposición en la pantalla, las relaciones entre componentes y la navegabilidad entre ellos.

    a. **Consistencia:** Respeta la visión de una consistencia interna, donde las convenciones y reglas son aplicadas a todos los elementos de la interfaz gráfica del usuario. Destacar su consistencia con el mundo real, respetando las convenciones y reglas consecuentes con la experiencia y percepciones del usuario con otras aplicaciones.

    b. **Disposición en pantalla:** La disposición espacial del menú principal y el menú auxiliar en la pantalla, son prácticamente una organización estándar en este tipo de aplicaciones. El menú principal, situado en el panel izquierdo y el menú auxiliar, situado en un panel superior al inicio de la pantalla, habilita un panel central como zona de trabajo. Con esta organización ayudará a que la aplicación sea más inteligible y menos saturada.

    c. **Tipografía:** Los componentes principales son el tipo de letra, estilos y técnicas. Las letras serán legibles, claras y singulares. Se respetará el uso de mayúsculas, minúsculas y demás símbolos ortográficos.

    d. **Color:** Esta propiedad es utilizada como una potente herramienta de comunicación si se usa correctamente. Los colores escogidos en el

sistema poseen una connotación añadida. El color negro y gris, informa sobre acciones comunes, el color azul representa información de interés para el usuario. Los tonos naranjas, representa información de alerta y consciencia por parte del usuario y finalmente el color rojo, representa una información crítica de conciencia inmediata.

- **Personalización:** En el caso el cual el usuario no se sienta cómodo con la organización predefinida por la aplicación por parte de la interfaz gráfica, este puede ocultar y redimensionar algunos elementos.

A continuación se mostrará una ilustración del menú principal de la aplicación, junto a un estado concreto del espacio de trabajo.

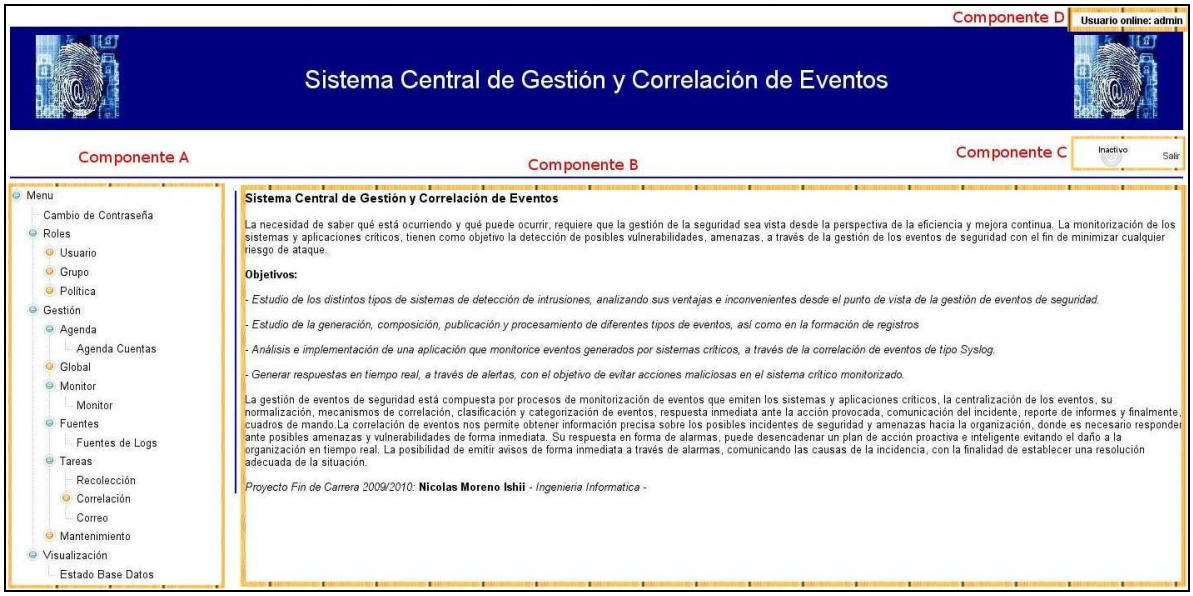


Ilustración 11 Pantalla inicial del sistema

La razón organizativa de las partes de la interfaz es como sigue.

- **Componente A:** Zona o sección de menú de las diferentes funcionalidades del sistema. Su estructura es en forma de árbol, cuyos nodos están diferenciados en dos colores. Color naranja indica que existen opciones por expandir, y el color azul indica que todas las opciones están presentes. Los nodos que forman el árbol pueden ser visibles o no.

- **Componente B:** Espacio de trabajo donde el usuario del sistema especifica, configura o ejecuta órdenes a la aplicación. La organización de los componentes gráficos se estructuran de forma homogénea e intuitiva.

- **Componente C:** Zona de finalización de la sesión actual e información del estado en el que se encuentra la conexión con el servidor. El componente

gráfico de "estado de conexión", informa del estado de la comunicación con el servidor, a través de un texto informativo. Los mensajes de estado de conexión, van acompañados con colores intuitivos, como son, el azul, naranja y rojo, coincidiendo con los estados correctos, alerta y desconexión respectivamente.

- **Componente D:** Zona informativa del usuario actual previamente autentificado.

3.4.2 Diseño de la base de datos

Para resolver el problema de la persistencia de la información, se ha decido utilizar una combinación de tecnologías. Se ha usado el lenguaje de etiquetado extensible XML para estructurar, almacenar e intercambiar información. La otra tecnología usada, es una base de datos clásica basada en la tecnología "Hyperthreaded Structured Query Language DataBase" (HSQLDB), por ser un gestor de bases de datos liviano y de libre distribución, fácil manejo y con una fiabilidad probada. Las gestiones administrativas de la aplicación utilizan la tecnología de XML, a través de SAX, como parseador del documento. El almacenamiento de eventos provenientes de las fuentes de recolección, se utilizado un gestor de base de datos HSQLDB, ideal para gestionar eventos de seguridad a una alta velocidad y con altas frecuencias de tiempo en la recepción de eventos.

A continuación se comentará la estructuración de la información que se almacena de forma persistente a través de ficheros XML:

- **Persistencia de usuarios, grupos y políticas:** Se almacenan los datos necesarios, relacionados con la autenticación y gestión de permisos del sistema.

    - **Usuarios:** Almacena los datos relativos a los usuarios que pueden acceder al sistema. Los campos principales que conforman un usuario son, un nombre, una descripción, un nombre público, una contraseña, una política asociada y aquellos grupos en los que está asociado el usuario.

    - **Grupos**: Almacena los usuarios bajo una política determinada. Los campos principales que conforman un grupo son, un nombre, una descripción, un nombre de pantalla, un campo condicional si el grupo se puede borrar, un campo condicional si el grupo se puede editar, y una política asociada.

    - **Políticas**: Describe la configuración de permisos sobre los diferentes grupos. Los campos que forman una política son, un nombre de política, una descripción, un nombre publico a mostrar, un campo condicional si la política se puede borrar, un campo condicional si la política si es editable, una lista de permisos asociados a esa política y una lista de grupos asociados a la política.

- **Persistencia de agenda de cuentas:** Se almacenarán aquellos datos de la configuración de protocolos y servicios del sistema.

    - **Cuentas de correo:** Contiene información de usuarios asociados a la agenda, como son sus direcciones de correo electrónico y los parámetros de configuración de posibles servidores SMTP. Los campos principales que conforman una cuenta de correo son, un identificador único, una descripción, una dirección de correo, un nombre de usuario, un nombre y un apellido del titular de la cuenta, un campo condicional refiriéndose si es una cuenta del sistema o no.

    - **Cuentas de Telnet:** Almacena los datos de configuración necesarios para establecer una conexión SSH con un servidor remoto. Entre los campos más importantes que conforman una cuenta de SSH se encuentran, un identificador único de cuenta, una descripción, un nombre de usuario, una dirección de red, una campo host haciendo referencia a la maquina remota a conectar, el puerto, el nombre de usuario de la maquina remota y la contraseña de acceso a la cuenta del sistema remoto,

    - **Cuentas de OpenJMS:** Contiene los datos necesarios para el correcto funcionamiento de las colas transaccionales de tipo OpenJMS. Se mostrará debido a su complejidad en la siguiente ilustración, los campos que conforman una cuenta de OpenJMS. La ilustración 12, especifica claramente los atributos de una cuenta OpenJMS.

```
<Cuenta EncriptarPwd="true" NombreUsuario="" Password=""
descripcion="Comunica los recolectores, con el formato comun"
esCuentaSistema="true" id="ColaRecoCorrelador" jmsPassword=""
jmsUser="" jndiContextFactory="org.exolab.jms.jndi.InitialContextFactory"
jndiCredentials="" jndiPassword="" jndiURL="vm:openjms"
jndiUser="" nombre="Cuenta de cola jms"
nombreClase="signware.notariolog.jms.CuentaJMS"
persistente="1" queueConnectionFactory="VMConnectionFactory"
tipo="jms" topicConnectionFactory=""/>
```

Ilustración 12 Tabla Cuenta OpenJMS

    - **Cuentas de Base de datos HSQLDB:** Almacena los datos necesarios para establecer la conexión con el gestor de base de datos. Se mostrará en la siguiente ilustración, los campos que conforman una cuenta de HSQLDB. En la ilustración 13, se muestra los atributos que conforman una cuenta de HSQLDB.

```
<Cuenta Alias="xdb" DatosConexion="jdbc:hsqldb:hsql://localhost/xdb"
Driver="jdbc:hsqldb:hsql" EncriptarPwd="true"
NombreUsuario="sa" Password="" Puerto="9001" TipoDriver="hsql"
TipoJDBC="" Url="localhost" tipo="db"
claseProtocolo="signware.notariolog.tools.db.ProtocoloBDHsql"
descripcion="Cuenta de acceso notario" esCuentaSistema="true"
nombre="Cuenta SA acceso a BDHsql" id="cuentaBD"
nombreClase="signware.notariolog.protocolos.db.CuentaBD" />
```

Ilustración 13 Tabla Cuenta HSQLDB

- **Persistencia de servicios de configuración global:** Almacena los datos necesarios para poder ofrecer los servicios de alertas.

    - **Servidor de SMTP:** Almacena los datos necesarios para el uso del proveedor de servicios SMTP. Como se puede observar, la ilustración 14 representa los atributos de una cuenta del protocolo SMTP.

```
<Protocolo SmtpHost="smtp.strato.com" SmtpIsAuth="true" SmtpPort="25"
nombreClase="signware.notariolog.protocolos.mail.ConfigCorreo"
SmtpPwd="          " SmtpUser="nmoreno@signware.es" componente="ProtoSMTP"
descripcion="Servidor SMTP strato" gestor="GestorConfigProtocolos"
id="strato" idFicheroVisual="ProtoSMTPPre.xml" nombre="strato"
systemMailAddress="nmoreno@signware.es" tipo="ProtoSMTP"/>
```

Ilustración 14 Tabla Protocolo SMTP

- **Persistencia de fuentes de logs:** Almacena los datos de configuración que describen una fuente. En la ilustración 15, podemos observar los atributos de configuración de una fuente.

```
<FuenteLog componente="FuentesLog" descripcion="Syslog" id="Syslog" nombre="Syslog"
nombreClase="signware.notariolog.fuentelog.FuenteLogSyslog" tipo="Syslog">
<Parseador id="Syslog1"/>
<InfoAplLogs>
<InfoAplLog aplicacionFuente="Syslog" maquinaFuente="192.168.2.199" puerto="1514"/>
</InfoAplLogs>
</FuenteLog>
```

Ilustración 15 Tabla Fuente Log

- **Persistencia de tareas:** Las tareas se configuran en documentos xml independientes dependiendo de su función particular.

    - **Tarea de recolección:** Almacena la información necesaria para poder configurar el recolector, describen las fuentes recolectadas. La ilustración 16, representa los atributos que forman la configuración de un recolector de fuentes.

```
<Recolector auditoria="true" componente="Recolector"
descripcion="Tarea Recoleccion" ejecutable="0"
gestor="GestorRecolectores" id="recosyslog3"
idFicheroVisual="RecolectorSyslogPre.xml" nombre="Recolector syslog"
nombreClase="signware.notariolog.recoleccion.RecolectorSyslog"
principal="false" tipo="RecolectorSyslog">
<RefFuentesLog>
        <RefFuenteLog activarCorrelacion="true" activarNotario="true"
        id="syslog4" tipoFuenteLog="Syslog"/>
</RefFuentesLog>
<Colas>
        <Cola id="1" idColaCorrelador="ColaRecoCorrelador"
        tipo="correlador"/>
</Colas>
        <Propiedades debugPaquetes="false" dirIP="192.168.2.199"
        puertoUDP="1514"/>
</Recolector>
```

Ilustración 16 Tabla Tarea Recolección

- **Tarea de correlación:** Almacena la información necesaria para poder configurar el correlador, reglas y alertas asociadas. Como podemos observar en la ilustración 17, las etiquetas y atributos que conforman una tarea de correlación.

```
<TareaCorrelador auditoria="true" componente="Correlador"
descripcion="Correlación interceptación auditable"
ejecutable="0" gestor="GestorCorrelacion" id="Intercep"
idFicheroVisual="TareaCorreladorPre.xml" nombre="Interceptacion"
nombreClase="signware.notariolog.tareas.correlacion.TareaCorrelacion"
principal="false" tipo="TareaCorrelador">
<Propiedades debug="false" idColaCorrelador="ColaFoCoCorrelador"
        idproveedorEsper=""/>
<Correlacion>
<Reglas>
        <Regla id="LEA_Ausente" idFicheroVisual="ReglaCorreladoraPre.xml"
        query="select * from Evento.std:unique(numeroDeSecuencia) e1,
        Evento.std:unique(numeroDeSecuencia) e2 where (not e1.LEA=null)
        and (not e2.mensaje=null) and
        not (FuncionesTexto.contieneTexto(e2.mensaje, e1.LEA))">
        <Alerta>
                <EnlaceAlerta destinatario="nmorenoi@gmail.com"
                gestor="GestorCorreo" idContexto="11" idTar=""
                tipo="contextualizado"/>
        </Alerta>
        </Regla>
</Reglas>
<Fuentes>
        <Fuente eventoClase="signware.formatoLogComun.Syslog"
        protocolo="LogFormatoComun"/>
</Fuentes>
</Correlacion>
</TareaCorrelador>
```

Ilustración 17 Tabla Tarea Correlación

- **Tarea de monitorización:** Almacena la información necesaria para poder poner en marcha las tareas de recolección y correlación. La ilustración 18, representa la estructuración de la información necesaria para configurar una tarea de monitorización.

```
<TareaMonitor auditoria="true" componente=""
descripcion="Tarea Monitor" ejecutable="O"
gestor="GestorMonitor" id="tareaMonitor"
idFicheroVisual="TareaMonitorPre.xml" nombre="TareaMonitor"
nombreClase="signware.notariolog.tareas.monitor.TareaMonitor"
principal="true" tipo="TareaMonitor">
        <TareasMon>
                <Recoleccion>
                        <EnlaceRecoleccion id="recosyslog3"/>
                        <EnlaceRecoleccion id="recoSSH"/>
                </Recoleccion>
                <Correlacion>
                        <EnlaceCorrelacion id="tareaCorrelar12"/>
                </Correlacion>
        </TareasMon>
</TareaMonitor>
```

Ilustración 18 Tabla Tarea Monitor

En lo referente a la gestión y almacenamiento de eventos, se crea una tabla dentro de la base de datos HSQLDB, acorde a la estructura de eventos de tipo Syslog, representada en la ilustración 19:

severidad	recurso	fecha	maquina	aplicacion	pid	mensaje
6	16	mar 19 may 21:25:20 CET	SignWare	Test		May 19 11:33:10 worklap pcscd: Control RxBuffer:
6	16	mar 19 may 21:25:22 CET	SignWare	Test		May 19 11:33:10 worklap pcscd: ifdhandler.c:924:IFDHPowerICC() lun: 0, action: PowerDown
6	16	mar 19 may 21:25:24 CET	SignWare	Test		May 19 11:33:10 worklap pcscd: ifdhandler.c:924:IFDHPowerICC() lun: 0, action: PowerUp

Ilustración 19 Tabla de Syslog

- **Prioridad**: Es un conjunto de números formado por las propiedades de recurso y severidad, aunque su significado suelen seguir una convención entre clientes y servidores.
    - **Recurso**: El tipo de apartado responsable de haber generado el mensaje.
    - **Severidad**: Establece la importancia del mensaje.

- **Fecha:** Tiempo o fecha en la que se ha registrado el evento.
- **Máquina:** Hostname o dirección IP de la máquina generadora del evento.
- **Aplicación:** Aplicación o programa que ha generado el evento.
- **Pid:** Identificador único del proceso o aplicación, dentro del sistema operativo.
- **Mensaje:** Descripción detallada del evento registrado.

3.4.3 Arquitectura software

En este apartado se describe la arquitectura software del sistema, a través de diagramas UML de despliegue, diagramas de clase y diagramas de secuencia. La ilustración 20 muestra el diagrama de despliegue, en el que los clientes del interactúan con la aplicación a través de un navegador Web. Cada cliente accede vía HTTPS al un servidor Web donde se aloja el sistema. El servidor Web interactúa con el servidor de base de datos y con los ficheros XML, con el fin de gestionar los datos de configuración transmitidos desde el cliente o los eventos de seguridad recolectados de las diferentes fuentes, como se puede observar en la ilustración 20, en el diagrama de despliegue.

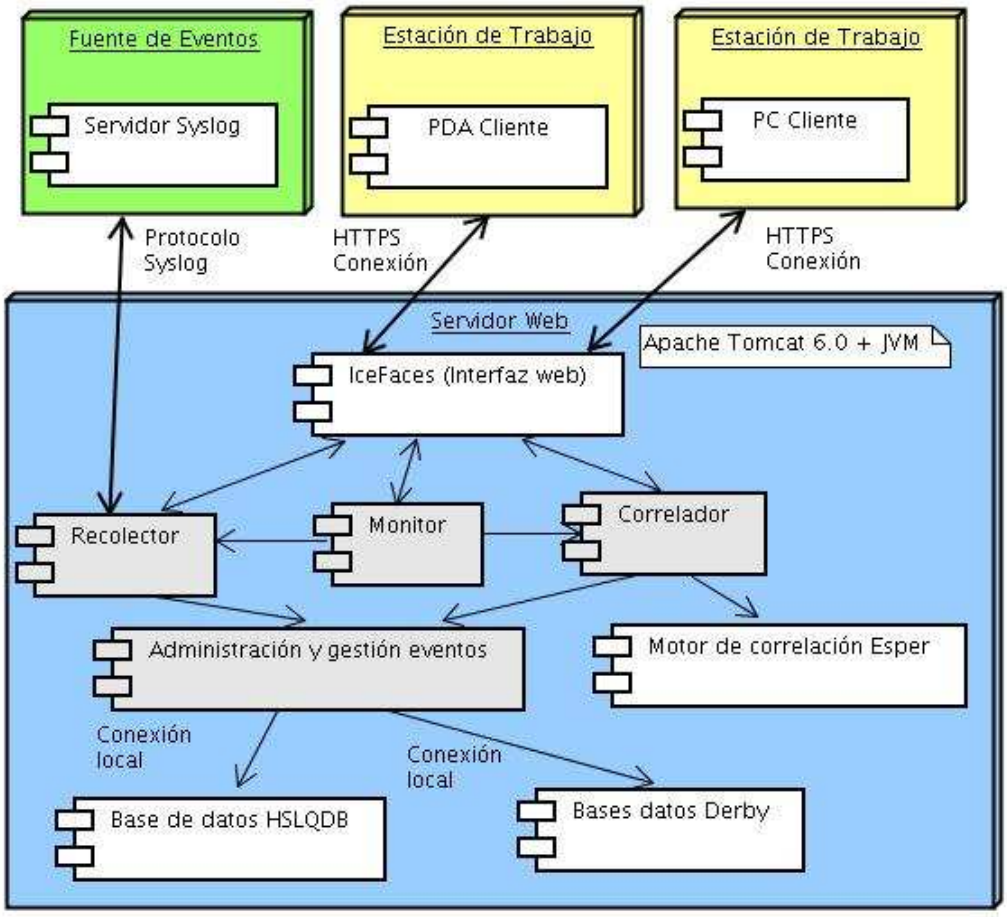


Ilustración 20 Diagrama de Despliegue

La descripción de los nodos del diagrama de despliegue es como sigue:

- **Estación de trabajo PC Cliente**: Se trata de un ordenador personal que contenga un navegador Web y realice la función de cliente interaccionando con el sistema.

- **Estación de trabajo Cliente**: Se trata del dispositivo que contenga un navegador Web y realice la función de cliente interaccionando con el sistema.

- **Servidor Web**: Aloja la aplicación Web dentro del sistema operativo y ofrece la accesibilidad comunicativa con el cliente y demás componentes.

Dentro del nodo Servidor Web tenemos la siguiente funcionalidad:

- **IceFaces:** Es el paquete que procesa las peticiones generadas por los clientes y se los envía a los "backbean", clases intermedios entre las peticiones gráficas y el modelo de negocio de la aplicación.

- **Monitor:** Representa aquella entidad que organiza y coordina la ejecución de la monitorización de eventos y su procesamiento.

- **Recolector:** Es la representación de todas los componentes que realizan la operación de obtener la información necesaria de las fuentes configuradas.

- **Correlador:** Organiza y administra las reglas y alertas de correlación comunicándose con el motor de correlación en su ejecución.

- **Motor de Correlación:** Se trata del procesamiento y correlación de eventos recibidos a través de la configuración realizada desde la aplicación.

- **Administración y gestión de eventos:** Engloban todos aquellos componentes que conforman la comunicación entre los eventos recolectados y su procesamiento. Además se encarga de almacenar la información de manera persistente.

- **Base de datos HSQL:** Representa la base de datos que contiene los eventos que han sido recibidos y procesados.

- **Base de datos Derby:** Representa la base de datos auxiliar de los eventos que han sido recién recibidos al sistema.

Una versión reducida del diagrama de paquetes y clases es el que se muestra en la ilustración 21, donde se muestran las partes principales de nuestro sistema.

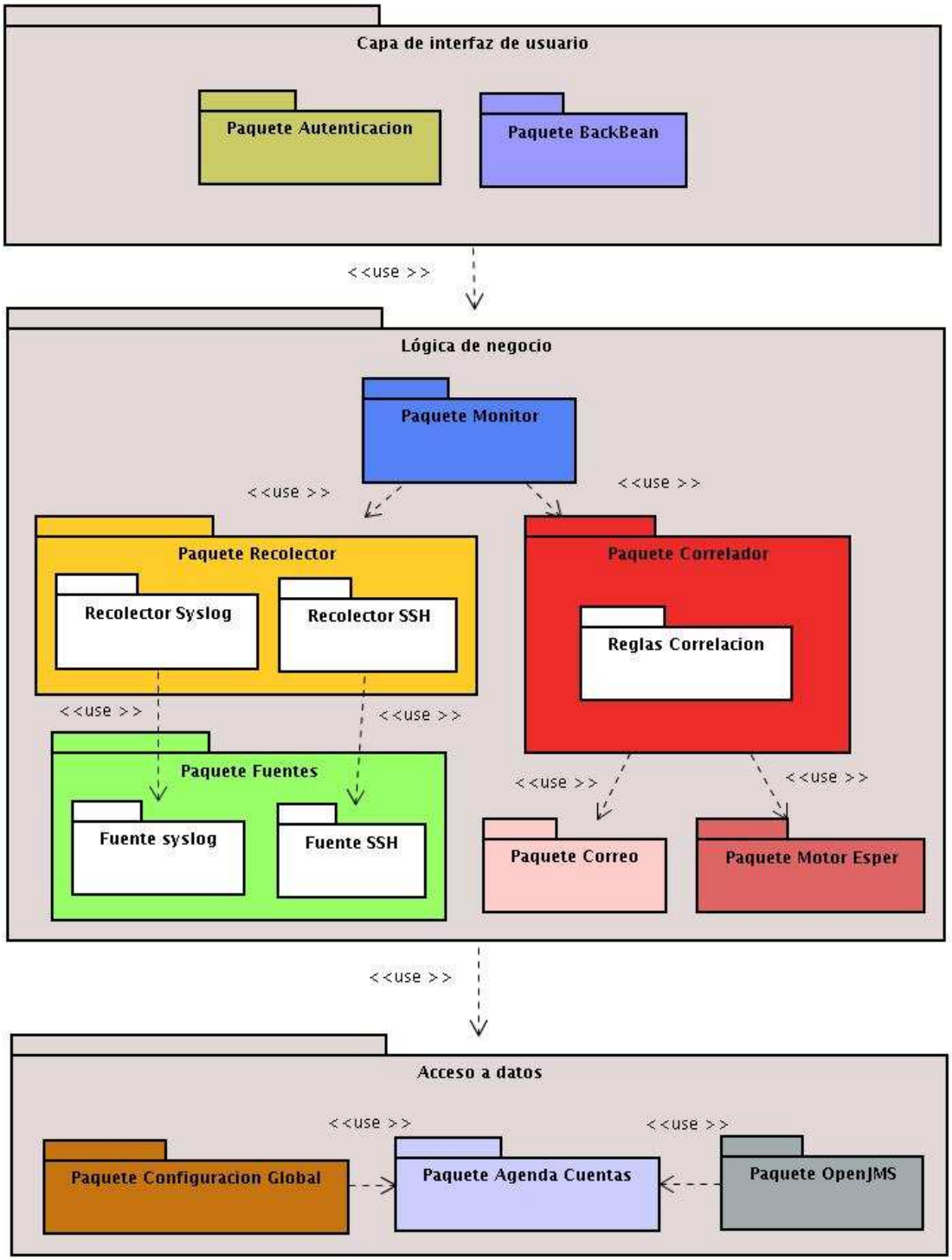


Ilustración 21 Diagrama de Paquetes Sistema

La descripción de las capas de la arquitectura es como sigue:

Capa de presentación: Contiene la parte de autentificación del sistema y administración de usuarios con sus políticas y el paquete "Backbean" encargado de las operaciones visuales para representar los componentes básicos del sistema.

Capa de modelo de negocio: Contiene la funcionalidad principal de la aplicación y está compuesta por los siguientes paquetes.

- **Paquete Fuente:** Representa a todas las clases que hacen posible la definición y gestión de una fuente. Entre las fuentes del sistema, podemos destacar los subpaquetes "syslog" y "SSH". Cada fuente definida, posee una relación asociativa con una cuenta, que define aquellos datos que son necesarios tener para establecer una conexión con la fuente remota.

- **Paquete Recolector:** Representa aquellas clases que son necesarias para poder gestionar y definir los recolectores de eventos. Estos recolectores son principalmente de tipo "syslog" y "SSH". Se establece una relación de asociación con una o varias fuentes del mismo tipo. También es necesario comentar, que una vez que se han recolectado los eventos de los sistemas remotos, estos se añaden a una cola de tipo OpenJMS, por lo que existe una relación con el paquete OpenJMS.

- **Paquete Correlador:** Representan las clases que son necesarias para poder configurar "reglas" y gestionar las alarmas de estas. Las reglas se manejan desde el subpaquete "reglas de correlación". Las alarmas son gestionadas en el paquete de "correo", por lo que el paquete correlador está íntimamente relacionado con el paquete de correo. Todos los eventos recibidos se leen de una cola OpenJMS, por lo que acceder a este servicio, es necesario establecer una relación de asociación con el paquete OpenJMS.

- **Paquete Correo:** Incluye todas aquellas clases que son necesarias para poder configurar y ejecutar un envío de correo electrónico. Esta tarea de correo, puede ser utilizada de manera independiente o asociada a una alerta de una tarea de correlación. La tarea de correo puede estar relacionada con un contacto de la agenda de cuentas, donde se dispondrá de toda la información necesaria para enviar un correo a la persona o entidad adecuada. Además posee una relación con el paquete de configuración global debido a la necesidad de disponer de una cuenta de SMTP para poder realizar el envío de un correo electrónico

- **Paquete Motor Esper:** Se incluyen todas las clases necesarias para el procesamiento de eventos recolectados de las fuentes remotas. Las clases que forman el motor de correlación poseen una relación asociativa con la tarea de correo, ya que este es el encargado de procesar eventos en relación con las reglas establecidas por la tarea de correlación. Las alarmas, son las encargadas de avisar

sobre el cumplimiento de una regla definida, ejecutan tareas de correo, por lo que existe una relación asociativa entre ellos.

- **Paquete Monitor:** Las clases que conforman el paquete monitor se encargan de gestionar y especificar, qué tareas de correlación y recolección, forman parte de un monitor. Este paquete posee una relación de uso, con los paquetes de recolección y correlación, de los cuales depende su identidad.

Capa de acceso a datos: Contiene los parámetros de configuración de las diferentes tareas del sistema, así como la cuentas de los usuarios y servicios. Además también contiene los eventos recolectados de las diferentes fuentes de datos.

- **Paquete OpenJMS:** En este paquete se incluyen todas las clases necesarias para poder establecer el servicio de OpenJMS, además de interconectar los módulos de recolección y correlación. Los eventos recibidos de la fuente, son introducidos en la cola y extraídos por las tareas de correlación. Para que un servicio de OpenJMS pueda establecerse, es necesario disponer de una serie de configuraciones básicas que están recogidas en una cuenta de servicio del paquete de agenda de cuentas.

- **Paquete Configuración Global:** Se agrupan todas aquellas clases que ayudan a gestionar aquellos servicios que son comunes en todo el sistema. Posee una relación con la agenda de cuentas, ya que es allí donde se encuentra toda la información global configurada de forma persistente. Realiza la función de almacenamiento de atributos necesarios para el correcto funcionamiento de la aplicación..

- **Paquete Agenda Cuentas:** Engloba todas las clases necesarias para poder gestionar todas las cuentas del sistema. Ofrecen principalmente un servicio de información agrupada y ordenada según sea un servicio o protocolo del sistema.

A continuación, se muestran algunos diagramas de clases que describen las relaciones de estas clases dentro de los paquetes mencionados anteriormente.

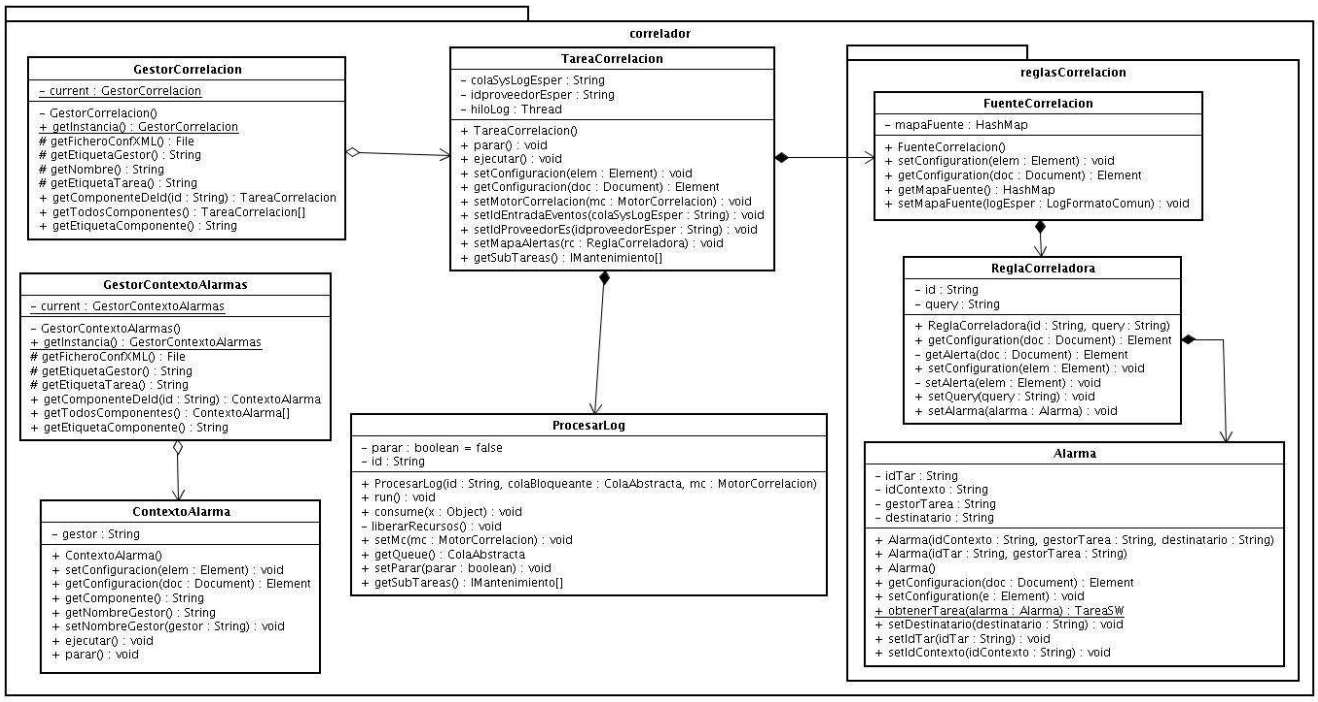


Ilustración 22 Diagrama de Clases Paquete Correlador

A continuación se describen, las clases que conforman el paquete “correlador”, representadas en el diagrama de clases, en la ilustración 22:

- **GestorCorrelacion**: Esta clase se encarga de gestionar las operaciones básicas de carga y salvado de datos, referentes a las tareas de correlación de forma persistente. Además gestiona la obtención de una tarea concreta de correlación a través de su identificador único.

- **TareaCorrelacion**: Es la clase que contiene toda la entidad de una operación de correlación. Posee el comportamiento de ejecutar y parar la tarea, además de almacenar las reglas de correlación y sus alertas, dentro de su estructura como atributos de clase.

- **ProcesarLog**: Es la clase que contiene la acción de consumición de eventos recibidos de la cola OpenJMS. Esta clase consume los eventos de la cola y los envía al motor de Esper para su procesamiento.

- **GestorContextoAlarmas**: Es la clase gestora que facilita las operaciones básicas de carga y salvado de mensajes contextuales.

- **ContextoAlarma**: Contiene aquella información necesaria para emitir un mensaje contextual de la situación de vulnerabilidad dentro de las posibles alarmas disponibles.

La descripción de las clases que conforman el subpaquete “reglasCorrelacion”, es la siguiente.

- **FuenteCorrelacion**: Es una clase que representa la estructura necesaria para organizar las reglas de correlación. Su estructura principal es un mapa hash.

- **ReglaCorreladora**: Las reglas correladoras agrupan las reglas de correlación, junto con las alarmas asociadas. Es una clase crítica y fundamental con respecto a las tareas de correlación.

- **Alarma**: Contiene los identificadores de las tarea de correo predefinidas, así como las contextualizaciones de los mensajes definidos en alarmas generadas por demanda.

La funcionalidad correspondiente al paquete de “MotorCorrelacion”, es como sigue en la ilustración 23.

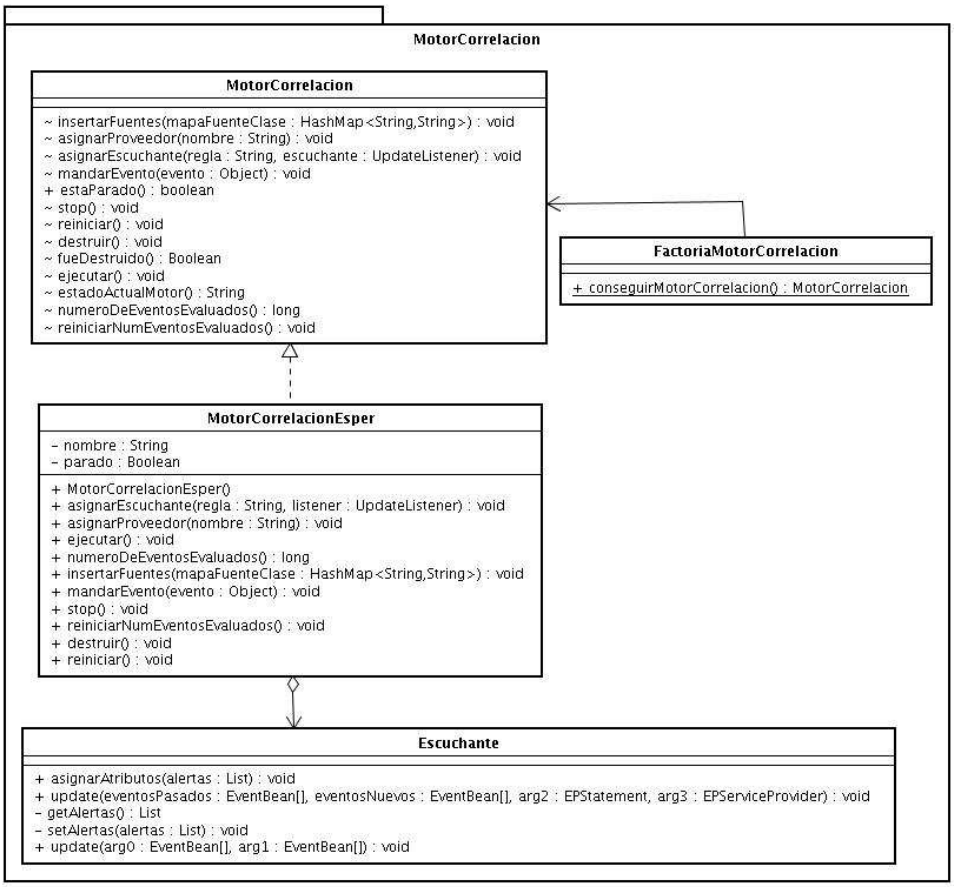


Ilustración 23 Diagrama de clases Motor de Correlación

- **FactoriaMotorCorrelacion**: Es una clase que sigue el patrón "Factory", destinado a devolver una instancia de un objeto, según el método de la clase que se invoque.

- **MotorCorrelacion**: Es una interfaz que proporciona únicamente el comportamiento a implementar por las demás clases. Posee aquellos métodos comunes en cualquier motor de correlación que se desee implementar.

- **MotorCorrelacionEsper**: Es la clase que implementa la interfaz "MotorCorrelacion" particularizado al motor Esper. En él se encuentra aquellas llamadas a los métodos del framework Esper y que permiten el procesamiento de eventos.

- **Escuchante**: Es el "listener" que se ejecuta de forma asíncrona, siempre que una regla se evalúe con éxito. Esta clase realiza la operación de ejecución de las tareas de correo, que fueron el resultado de las alarmas previamente configuradas.

Una vez finalizado con la parte de correlación de eventos, describimos el paquete de correo que representa la ilustración 24.

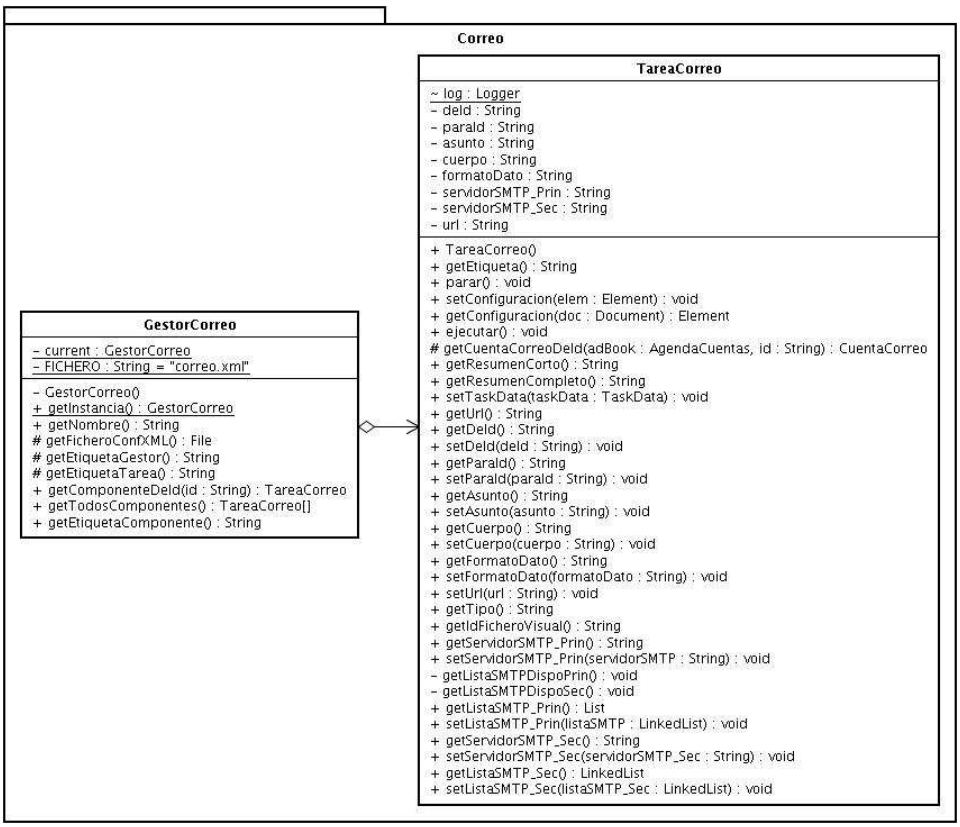


Ilustración 24 Diagrama de clases Tarea Correo

- **GestorCorreo**: Es la clase que administra y gestiona las operaciones básicas de las tareas de correo. Se encarga de almacenar y cargar las tareas configuradas, además de ofrecer aquella tarea de correo que necesita en un momento determinado, a través de un identificador único de la tarea.

- **TareaCorreo**: Es la clase que agrupa la información necesaria para mandar un correo electrónico. Como cualquier tarea del sistema, posee los métodos "ejecutar" y "parar", además de poseer una lista de los servidores de SMTP asociados.

Las clases que conforman el paquete "recolector", representados en la ilustración 25, se estructuran a través de un diagrama de clases de la siguiente forma:

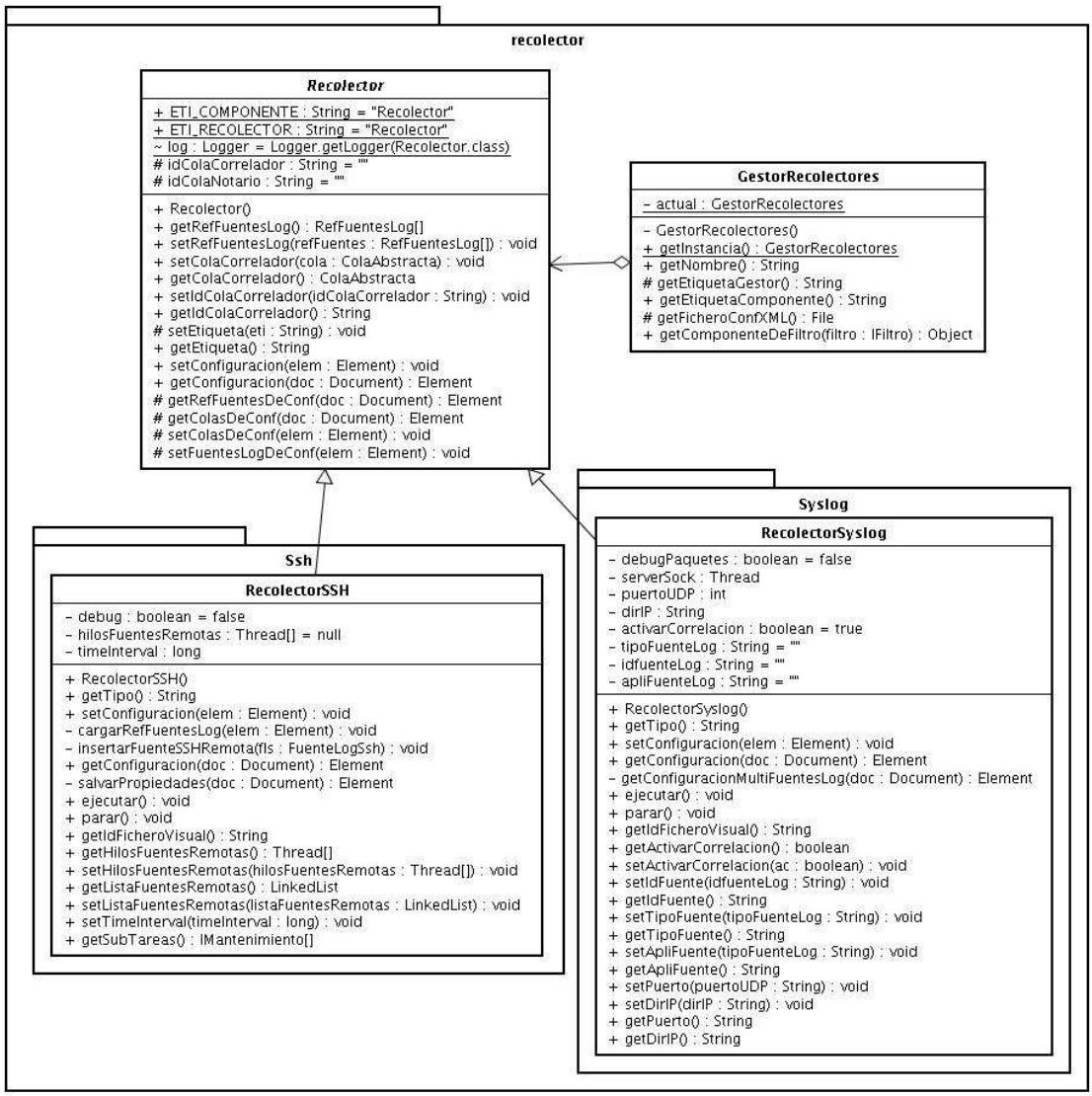


Ilustración 25 Diagrama de clases Recolector

En el paquete “recolector”, nos encontramos las siguientes clases y subpaquetes, responsables de la obtención de eventos de las máquinas previamente especificadas:

- **Recolector:** Es una clase abstracta, que representa los métodos comunes a todos los tipos de recolectores. Facilitando así de forma homogénea, el poder trabajar con los recolectores, a través de las clases gestoras.

- **GestorRecolectores:** Es la clase que administra y gestiona el uso de los recolectores. Además carga en memoria y almacena de forma persistente, aquellos recolectores que se han creado o se han configurado. Contiene métodos que permiten obtener un determinado recolector a través de su identificador único.

- **RecolectorSSH:** Es la clase que asocia las fuentes de SSH configuradas. Las fuentes se agrupan en uno o varios recolectores, estos pueden realizar operaciones de ejecución de recolección de eventos de sus fuentes

.

- **RecolectorSyslog:** Esta clase tiene asociada aquellas fuentes de tipo syslog. Como cualquier recolector, posee la capacidad de realizar las operaciones de ejecución de recepción de eventos, a través de las fuentes asociadas, así como la posibilidad de parar su recolección.

Las clases que conforman el paquete “fuentes”, tal y como muestra la ilustración 26, se encargan de obtener la información de los eventos que se reciben de los sistemas monitorizados, de forma estructurada y parseada:

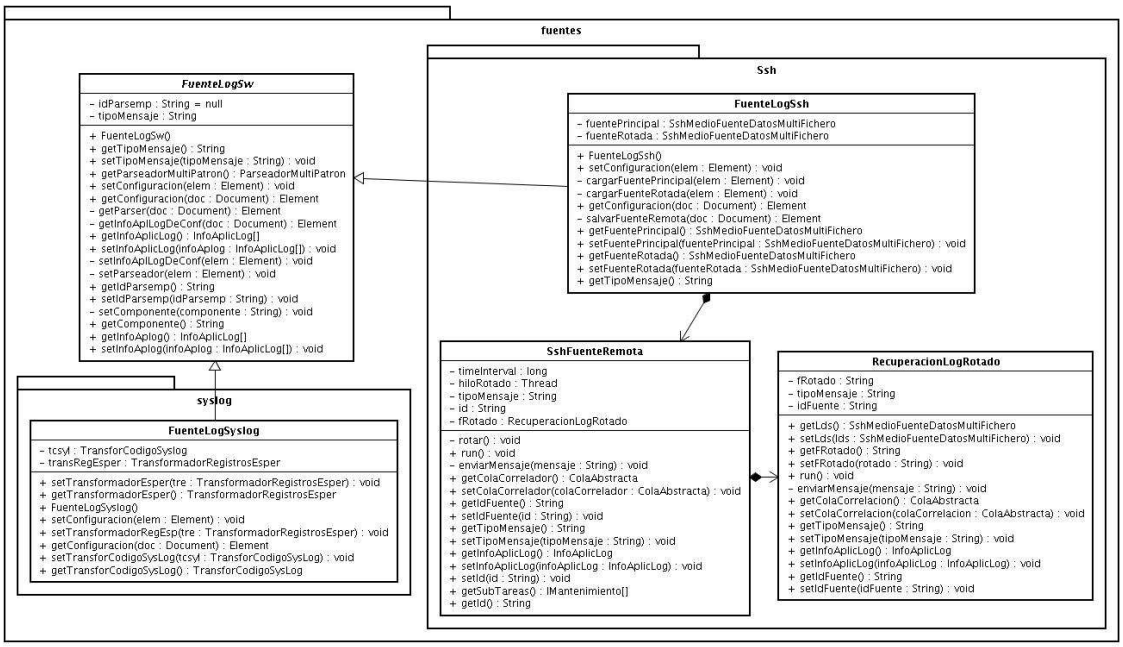


Ilustración 26 Diagrama de clases Fuentes

- **FuenteLogSW**: Es la clase abstracta que implementa todas las fuentes del sistema. Posee el comportamiento común a todas las fuentes de eventos que se puedan recolectar, ofreciendo una interfaz homogénea entre todas ellas.

- **FuenteLogSyslog**: Representa una fuente de tipo “syslog”, principalmente realiza la escucha a través de un puerto, donde los clientes del protocolo syslog le envian los eventos.

- **FuenteLogSSH**: Representa la fuente que recoge eventos de un fichero remoto a través del protocolo de SSH. Esta clase se encarga de establecer una relación entre la fuente principal y la fuente rotada. Esto es debido, a que los ficheros de logs realizan rotaciones de ficheros, cuyos eventos son necesarios recolectar.

- **SSHFuenteRemota**: Esta clase realiza la función de obtener los eventos de la fuente principal en tiempo real. Su mecanismo de acción es establecer una conexión SSH con el servidor y ejecutar un comando que le permita leer el fichero donde se escriba los eventos. Estos eventos una vez leídos se escribirán en la cola de eventos recolectados.

- **RecuperacionLogRotado**: Posee el mismo comportamiento que la clase “SSHFuenteRemota”, pero con la diferencia que procesa eventos del fichero rotado.

A continuación se comenta el paquete "monitor", representado por la ilustración 27, el cual se encarga de la coordinación de los componentes críticos de la aplicación.

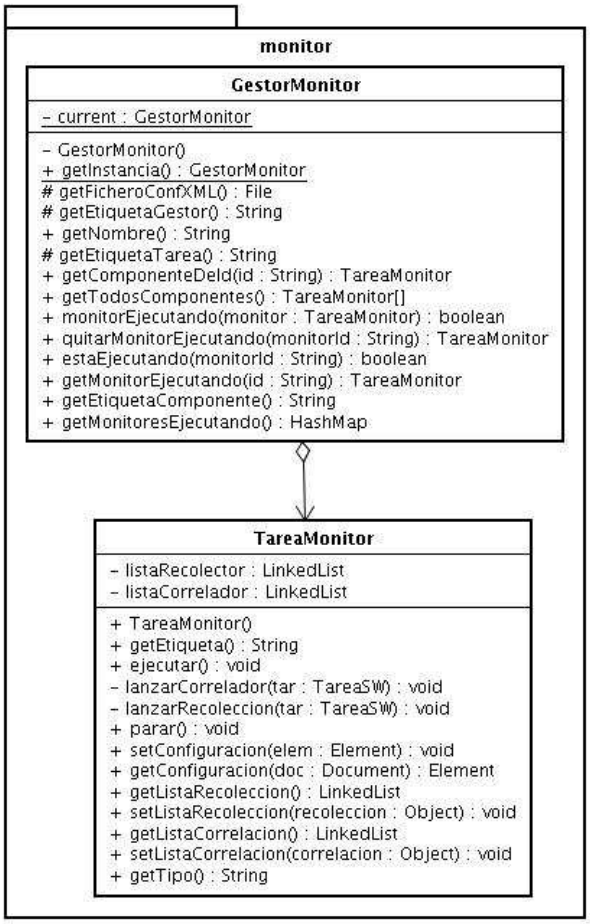


Ilustración 27 Diagrama de clases Monitor

- **GestorMonitor**: Es la clase gestora, encargada de mantener en memoria las tareas de monitor configuradas, y almacenarlas de forma persistente. Además ofrece la posibilidad de obtener la tarea de monitor elegida.

- **TareaMonitor**: Es una de las tareas más importantes del sistema, ya que engloba el conjunto de tareas y fuentes. Hay que destacar que el monitor posee dos listas donde almacena las tareas de correlación y las tareas de recolección asociadas al monitor. Como cualquier tarea configurada posee los métodos de "ejecutar" y "parar". En la ejecución de la tarea de monitor, está llamará al método ejecutar de la lista de tarea de correlación y recolección, provocando así una ejecución de sus subtareas.

El paquete “OpenJMS” representado en la ilustración 28, conecta la recolección de eventos con la correlación de los mismos. La descripción de las clases que implementan su funcionalidad es la siguiente.

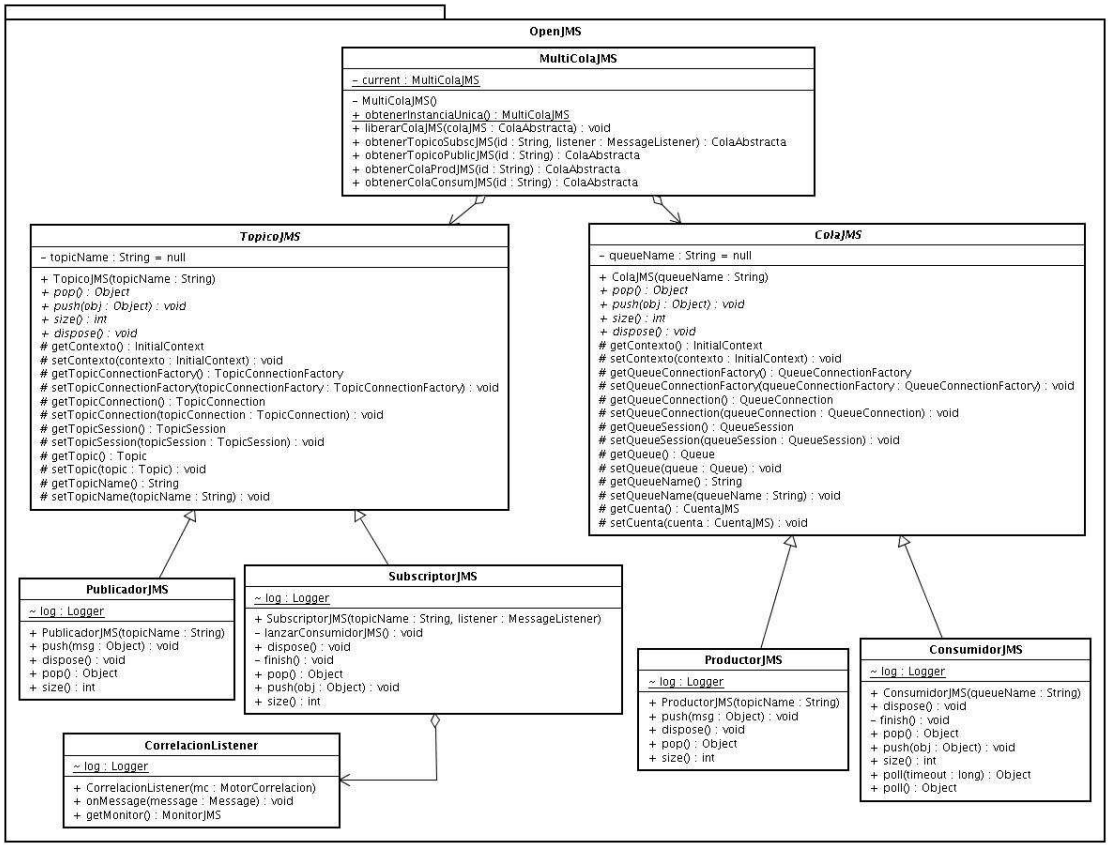


Ilustración 28 Diagrama de clases OpenJMS

OpenJMS es el servicio que interconecta las tareas de recolección y correlación a través de su estructura que implementa una cola de eventos. Este servicio es fundamental para el correcto funcionamiento de la aplicación. Lo conforman las siguientes clases:

- **MultiColaJMS**: Es una clase abstracta, que especifica el comportamiento necesario para poder gestionar una estructura de tipo cola. Se utiliza esta clase cada vez que se necesite un acceso a la cola. También posee la responsabilidad de liberar la cola.

- **TopicoJMS**: Es una clase abstracta que representa la abstracción de una cola asíncrona. A diferencia de los otros tipos de implementaciones, esta cola sigue el patrón “publicador – subscriptor”. También realiza la operación de inicializar la cola a través de una cuenta OpenJMS definida.

- **PublicadorJMS**: La clase representa la parte publicadora de un evento en el servicio OpenJMS. Este evento llegaría de forma asíncrona a todos aquellos objetos subscritos al servicio.

- **SubscriptorJMS**: Es la clase que representa la parte pasiva del patrón "publicador – subscriptor". Esta clase determina que "listener", es responsable de la lógica que se ejecuta cuando se produce un evento.

- **CorrelacionListener**: Representa el "listener" que será invocado al publicar un nuevo evento en el servicio OpenJMS. Esta clase es responsable de realizar un posible agrupamiento de eventos.

- **ColaJMS**: Es una clase abstracta que representa el comportamiento clásico de una estructura de tipo cola síncrona. También realiza la operación de inicializar la cola a través de una cuenta OpenJMS definida.

- **ProductorJMS**: Es la clase que representa el productor de eventos a la cola. Se utiliza por los recolectores del sistema, una vez que se obtiene un evento de las fuentes monitorizadas. También es utilizada por las tareas de correlación, en el envío de eventos hacia la cola que comunica con el motor de correlación.

- **ConsumidorJMS**: Esta clase representa el consumidor de la cola de eventos. Es utilizada por el correlador en el momento de procesar los eventos enviados por los recolectores. También es utilizado por el motor de correlación para recibir los eventos consumidos por la tarea de correlación y enviados el motor Esper.

Finalmente, los diagramas de secuencia que consideramos más relevantes son los siguientes. La ilustración 29, representa las operaciones que se realizan durante la configuración de una tarea de correlación.

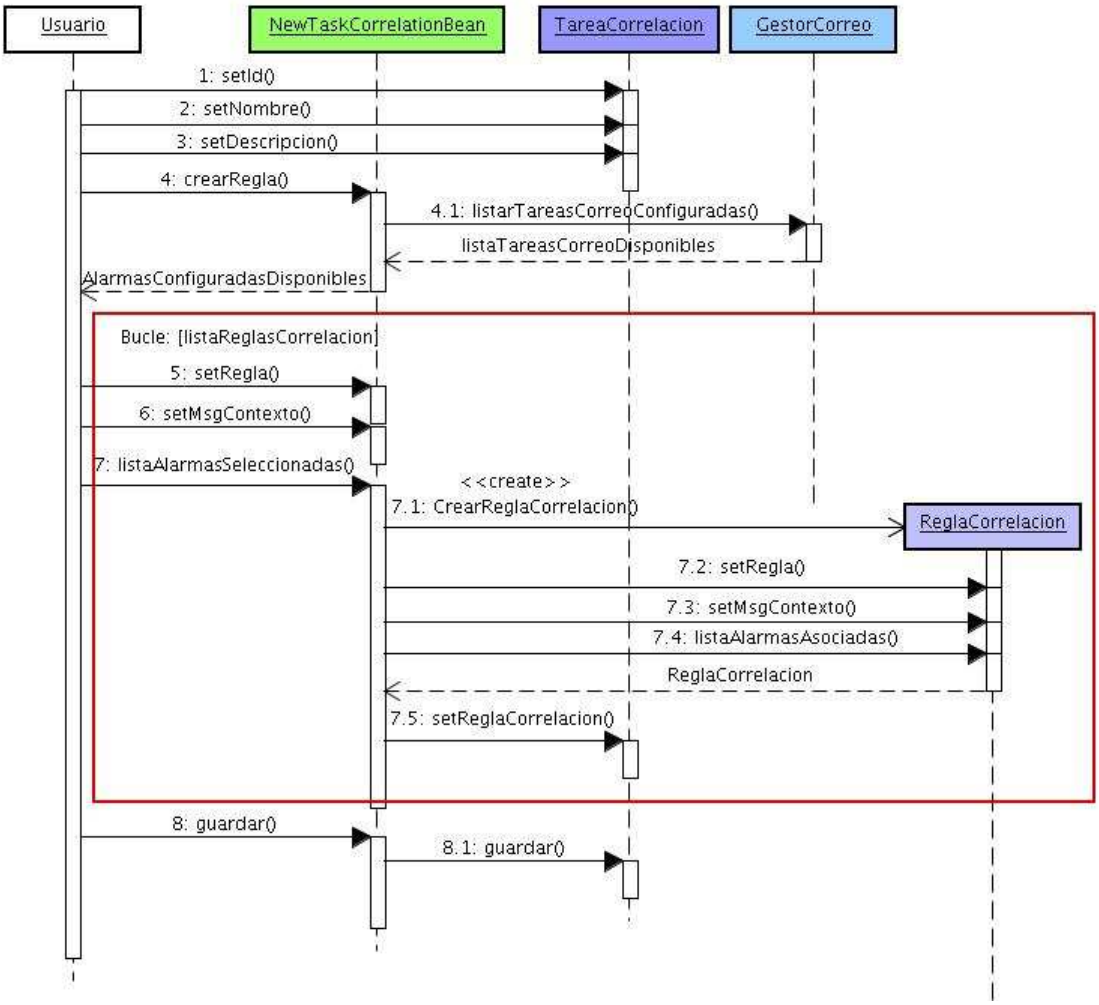


Ilustración 29 Diagrama Secuencia Crear Tarea Correlación

El técnico del sistema desea realizar la operación de creación de una tarea de correlación. Para crear cualquier tarea, el usuario debe definir un identificador único de la aplicación (1), un nombre (2) y una descripción de la tarea (3). Una vez rellenados esos campos obligatorios, es necesario que el técnico defina las reglas de correlación y las alertas asociadas a las reglas. El sistema propondrá una serie de alarmas disponibles, que son tareas de correo actualmente configuradas en el sistema (4). Para definir una regla, el técnico definirá la regla de correlación en el lenguaje EPL (5), además si este lo desea, puede añadir un mensaje en contexto con la regla, a través de una tarea de correo creada por demanda, lo incorporará a la tarea de correlación (6). El sistema dará la opción de elegir las tareas de correo ya configuradas en forma de plantilla, por si el técnico deseara incorporarlas como alarmas en la tarea de correlación. Este conjunto de alarmas de correo contextualizadas y creadas por

demanda forman el conjunto de alarmas seleccionadas que se envían al "backbean" (7). Por cada configuración que realice el técnico sobre las reglas y sus alarmas, el "backbean" crea una regla (7.1), asigna la especificación EPL a la regla (7.2), asigna el contexto del mensaje de la tarea de correo asociada a la regla (7.3) y la lista de alarmas asociadas (7.4), donde finalmente el "backbean" asocia la regla de correlación configurada, a la tarea de correlación creada (7.5). Una vez realizadas todas las definiciones de las reglas, se almacena la configuración de la tarea de forma persistente en el sistema (8).

A continuación, la ilustración 30, se describe el proceso que sigue el sistema en el momento de ejecutar un monitor determinado ya configurado, por el usuario técnico de la aplicación.

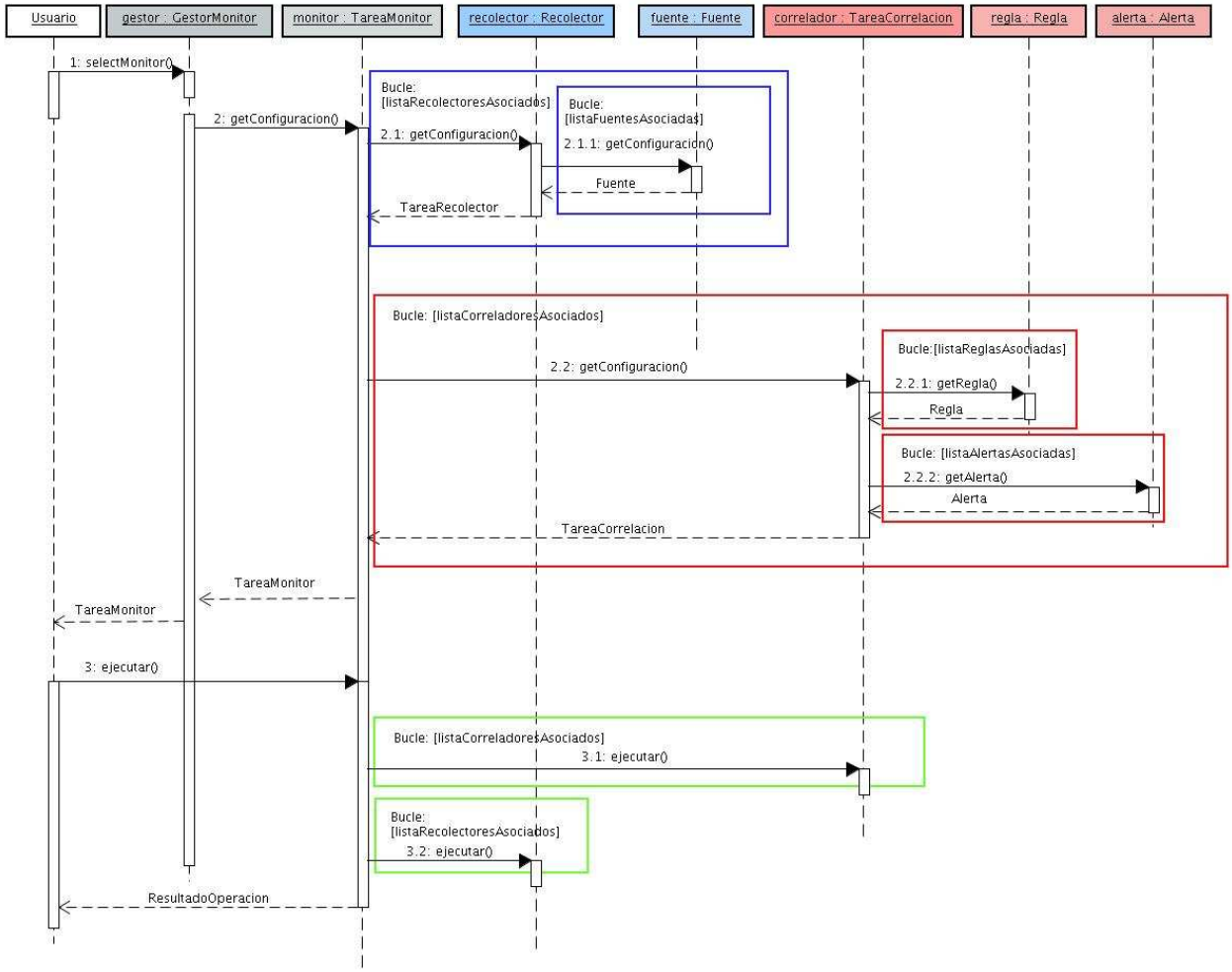


Ilustración 30 Diagrama Secuencia Ejecución Monitor

La operación que realiza el técnico en el momento de poner en funcionamiento un monitor, empieza por la selección del monitor de la lista de monitores ya configurados (1). El gestor de monitores, a través del "backbean" que envió la elección

del usuario, identifica el monitor seleccionado y lo carga en memoria (2). Para que un monitor esté en memoria, es necesario leer del fichero de configuración las especificaciones de la tarea de recolectores asociados (2.1), donde cada recolector tiene asociado una serie de fuentes, que son leídas de igual forma desde el fichero de configuración de fuentes y cargadas en memoria (2.1.1). Una vez finalizada la carga en memoria de todos los recolectores, es necesario almacenar en memoria las tareas de correlación asociadas al monitor (2.2). Cada tarea de correlación está compuesta por una o varias reglas (2.2.1) y una o varias alertas (2.2.2). Finalmente, con los datos en memoria, se ejecutan (3) las tareas de correlación (3.1) y recolección (3.2).

3.5 Implementación

3.5.1 Aplicación Web

A continuación se detallan las partes más relevantes de la aplicación pertenecientes a los paquetes que contienen el código de la fuente SSH, tarea de recolección SSH, tarea de correo, tarea de correlación y tarea monitor.

El paquete que engloba las clases que realizan la recolección de eventos de tipo Syslog a través de un fichero, son recogidos gracias al protocolo SSH y al editor de texto “less”. La clase “**SSHFuenteRemota.java**”, se encarga de establecer una conexión SSH con una máquina remota. Una vez establecida la conexión correctamente, abre el fichero remoto y lee línea a línea. Mediante el paquete “**j2SSH-common.jar**”, es posible establecer una sesión remota con la máquina y poder ejecutar comandos de shell remotamente. La salida estándar de la máquina remota, es decir las líneas de texto leídas del fichero remoto, son recogidas por la sesión establecida e introducidas en una cola. Cada elemento de la cola corresponde a una línea impresa por la salida estándar de la “shell” remota. Se visualiza cada línea del fichero remoto que procese el editor “less”, interpretándose así cada línea como un evento.

Cada evento que es obtenido y procesado, se añade a una lista llamada “cacheList”. La lista caché está vacía al inicio de la ejecución. Esta lista será revisada cada vez que se recojan eventos de la cola, por lo que al inicio, todos los eventos serán nuevos, pero en la segunda llamada a la cola, los primeros eventos obtenidos pasarán a ser eventos antiguos. Los eventos antiguos se ignoran y no son procesados nuevamente. Aquellos que no estén en la caché se enviarán a la cola del recolector y se añaden a la lista de la caché.

Otra cuestión a tener en cuenta, es el procesamiento del fichero remoto. La lectura se realiza de forma secuencial. No se contempla que la fuente pueda ser modificada en alguna línea que haya escrito y leído anteriormente. Para enviar el evento a la cola transaccional del recolector, es necesario que el evento leído cumpla la condición que la línea leída no sea un evento ya contenido en la caché y además el

tamaño los bytes leídos totales hasta el momento del fichero, sea distinto al leído en la anterior iteración. Si el tamaño de los bytes no cambia, no se han escrito nuevos eventos.

En la ilustración 31, se observa que cada vez que se ha leído un bloque de eventos de la cola, nuestra aplicación cierra la sesión, conservando la conexión y esperando un intervalo de tiempo determinado. Esta parada es necesaria para que se puedan producir nuevos eventos en el fichero, además de descargar los recursos utilizados por el sistema en el procesamiento de eventos a través de SSH. Una vez cumplido el intervalo de tiempo asignado, el sistema de procesamiento de eventos se pone nuevamente en funcionamiento. En primer lugar, se comprueba que la conexión esté establecida. En ese caso se lanza nuevamente la sesión y se procesarán los nuevos eventos que se han escrito en el fichero remoto, de la misma forma anteriormente comentada.

```
public void run(){

  while(!parar && !Thread.interrupted()){
    DataReference dataRef=null;
    DataReference dataRefOld=new DataReference();
    long size=0; long sizeOld=0; boolean rotado=false; String newEvent=null;
    try {
        lds.update();
        if(!lds.isConnected()){Thread.sleep(500);continue;}
        lds.pararHiloLector();
        DataSourceResult log=lds.nextLogResult();
            if(log!=null)
                size=lds.getSize();
                   while(!rotado && !parar){
                        while(log!=null && !parar){
                                if(log!=null){
                                  newEvent=log.getValue().toString();
                                  if(size!=sizeOld && !estaEnCache(newEvent)&& !parar){
                                    this.enviarMensaje(log.getValue().toString());
                                    this.cacheList.addLast(log);
                                    dataRefOld=dataRef;
                                   }
                                }
                                log=lds.next();
                        }
                        sizeOld=size;
                        lds.dispose();
                        Thread.sleep(this.getTimeInterval());
                        lds.update();
                        if(!lds.isConnected()){Thread.sleep(500);continue;}
                        lds.pararHiloLector();
                        log=lds.nextLogResult();
                        size=lds.getSize();
                           if(sizeOld>size){
                                this.rotar();
                                rotado=true;
                                break;
                           }
                   }
    }catch(Exception e){this.liberarRecursos();}
}
```

Ilustración 31 Implementación fuente SSH

Una cuestión crítica, es el problema de la rotación de un fichero. Un fichero rota cuando su capacidad de almacenamiento ha sido superada. El proceso de rotación generalmente es el cambio de nombre del fichero y la creación de un nuevo fichero. El fichero anterior se denominará “fuente rotada” y el nuevo fichero “fuente remota o principal”. La gestión del problema en el momento del procesamiento es bastante complejo. Puede ocurrir que los eventos que se escriban en las últimas filas del fichero, no han sido procesados en el momento de la rotación, ya que el hilo principal que gestiona la lectura de eventos está en el tiempo de espera. Cuando procesadores hilo que procesa las fuentes remotas se inicia de nuevo, el fichero procesado podría haber realizado la rotación antes. Si es así, nos encontraríamos ante la situación en la que la cantidad de bytes leídos hasta el momento en el fichero antiguo, fuera mayor al que contiene el fichero nuevo creado, producto de la rotación. Para solucionar este problema, se ha decidido que cuando el procesador de fuentes remotas inicialice su procesamiento se compruebe que el tamaño del fichero sea igual o mayor, al fichero que se estaba procesando antes del tiempo de espera. Si el número de bytes es menor, entonces podemos saber que ha producido una rotación y podemos recuperar los últimos eventos no procesados.

Para poder procesar los últimos eventos, es necesario establecer una nueva conexión con una fuente distinta pero en la misma máquina. El mecanismo es el mismo que para una fuente principal, con la diferencia que en esta ocasión, solo se buscan los eventos que no están en la caché de la fuente principal en el momento de la parada. Se procesará línea a línea el fichero rotado, y se envían a la cola aquellos eventos que no estén en la caché de la fuente principal antigua. Este proceso es bastante costoso, pero hay que tener en cuenta, que las rotaciones normalmente se hacen pocas veces al día o a la semana. Una vez solucionado el problema de la rotación, la caché se limpia y el mecanismo de recolección de eventos remotos a través de SSH empieza de nuevo.

Una tarea recolector SSH, puede estar asociada a un número de fuentes SSH indeterminados Cada recolector SSH tiene una lista de fuentes remotas como atributo de clase. Cuando el recolector se configura, es necesario especificar un intervalo de tiempo de acceso a los eventos remotos. Este tiempo, se transmite a cada fuente SSH remota para que lo aplique en el mecanismo de procesamiento de eventos, anteriormente comentado. El recolector SSH es bastante sencillo, como se puede observar en la ilustración 32. Su objetivo es ejecución de todas las fuentes asociadas, cuando el monitor le mande la orden. Cada fuente realiza su proceso de obtención de eventos sobre los ficheros especificados remotos a través de su cuenta SSH.

```
public void ejecutar() {
        this.getEstatusEje().setFinish(false);

        try{
        //Almacenamos los hilos de los recolectores ssh remotos
        this.hilosFuentesRemotas=new Thread[this.getListaFuentesRemotas().size()];
        for(int i=0;i<this.getListaFuentesRemotas().size();i++){
                SshFuenteRemota sfr=(SshFuenteRemota)this.getListaFuentesRemotas().get(i);
                hilosFuentesRemotas[i]=new Thread(sfr,"SshFuenteRemota-"+i);
        }

        //Ejecutamas los hilos remotos
        for(int i=0;i<this.hilosFuentesRemotas.length;i++){
                this.hilosFuentesRemotas[i].start();
        }

         }catch(Throwable th){
                log.error("Error en ejecucion: fin RecoleccionSSH",th);
                this.getEstatusEje().setFailed(true);
                parar();
         }

}
```

Ilustración 32 Implementación ejecutar Tarea Recolector SSH

Con el fin de detener la obtención de las fuentes remotas, como se observa en la ilustración 33, el proceso es el mismo que de la ejecución. El array de hilos paralelos e idéntica a la lista de fuentes remotas, provocará una excepción de interrupción sobre las mismas. Los hilos se van parando uno a uno de forma ordenada y síncrona.

```
public void parar() {

        if(log.isDebugEnabled())
                log.debug("Inicio parar recolector ssh "+this.getId());

        try{
         if(this.hilosFuentesRemotas!=null && this.hilosFuentesRemotas.length>0){
          for(int i=0;i<this.hilosFuentesRemotas.length;i++){
            this.hilosFuentesRemotas[i].interrupt();
            SshFuenteRemota sfr=(SshFuenteRemota)this.getListaFuentesRemotas().get(i);
            sfr.setParar(true);
            this.hilosFuentesRemotas[i].join();
          }
         }
         MultiColaJMS.obtenerInstanciaUnica().liberarColaJMS(this.getColaCorrelador());
         MultiColaJMS.obtenerInstanciaUnica().liberarColaJMS(this.getColaNotario());
         log.info(RecolectorSSH.class.getSimpleName()+": "+this.getId()+" Parado");
         this.getEstatusEje().setFinish(true);

        }catch(Exception e){
                log.error("ERROR en parar el RecolectorSSH: ",e);
        }
}
```

Ilustración 33 Implementación parar Tarea Recolector SSH

La tarea de correlación, es la responsable de inicializar el motor de correlación introduciendo en él las reglas configuradas. Hay que destacar, que la tarea de

correlación posee como atributos de clase una instancia al motor de correlación Esper, un hilo encargado de consumir eventos de la cola de recolección, un mapa de reglas y un mapa de alertas.

En el momento de ejecución, la tarea de monitorización, ordena a la tarea de correlación que se ejecute invoca al método ejecutar. El método ejecutar de la tarea de correlación, tal y como muestra la ilustración 34, inicializa el objeto que implementa el hilo que procesa la cola de eventos recolectados, y se lo envía al motor de correlación Esper. El hilo recolector posee el nombre de ProcesarLog. Para inicializar la clase "ProcesarLog.java", hay que añadirle como parámetros al constructor, una referencia a la cola donde están los eventos recolectados y una instancia al motor Esper ya inicializado. Finalmente el hilo se ejecuta.

```
public void ejecutar(){

        try{
         this.getEstatusEje().setFinish(false);
         //convertido en evento
         cEspert = (ColaAbstracta)MultiColaJMS.obtenerInstanciaUnica().obtenerColaConsumJMS(colaSysLogEsper);
         //Consumir Logs

         ProcesarLog procesador= new ProcesarLog("Procesador de eventos",cEspert,motorc);
         this.listaSubtarea[0]=procesador;

         hiloLog = new Thread(this.listaSubtarea[0],"Hilo Correlacion");
         hiloLog.start();

         this.getEstatusEje().setCurrentStatus(IMantenimiento.TAREASW_EJEC);

         log.info("Tarea de Correlacion ejecutada correctamente "+this.getId());
         this.getEstatusEje().setFinish(false);

        }catch(Exception e){
                log.error(e.getMessage());
                this.getEstatusEje().setFinish(true);
                this.getEstatusEje().setCurrentStatus(IMantenimiento.TAREASW_ERROR);
                this.getEstatusEje().setFailed(true);
        }
}
```

Ilustración 34 Implementación ejecutar Tarea Correlación

La clase ProcesarLog, como se muestra en la ilustración 35, implementa un hilo para que consuma de la cola los eventos que se vayan añadiendo y los envíe al motor de correlación.

```
public void run() {
   try {
           this.estadoEjeTarea.setFinish(false);
           while (!parar) {
             this.getEstadoEjeTarea().setCurrentStatus(IMantenimiento.PROCESW_CORRELA_EJEC);
             consume(queue.pop());
           }
   }catch(InterruptedException ie){
           this.liberarRecursos();
           log.debug(ie.getMessage());
   }catch (Exception ex) {
           this.liberarRecursos();
           log.error(ex.getMessage());
           this.estadoEjeTarea.setFailed(true);
           this.estadoEjeTarea.setCurrentStatus(IMantenimiento.PROCESW_CORRELA_ERROR);
   }
}

public void consume(Object event) throws Exception{
    try{

      SyslogEsper sys=this.formatearEvento(event);
              mc.mandarEvento(sys);
              System.out.println(((MensajeLog)event).getMsg());
              cnt++;
    }catch (Exception ex) {
           this.liberarRecursos();
           log.error(ex.getMessage());
   }
}
```

Ilustración 35 Implementación ProcesarLog

La tarea de monitorización, asocia las tareas de recolección de eventos y las tareas de correlación. La clase tarea monitor posee dos listas de elementos de tipo tarea. Una lista es para las tareas de recolección y la otra lista es para tareas de correlación seleccionadas desde la parte gráfica. El método ejecutar de la tarea de monitor, como se presenta en la ilustración 36, lanza de forma ordenada en primer lugar, las tarea de correlación que tiene vinculadas, y en segundo lugar, lanzará las tareas de recolección. El orden de lanzamiento sigue un orden lógico, no tiene sentido que los recolectores se inicien antes, obteniendo eventos y el motor de correlación todavía no esté en marcha aún, aunque en ese caso, esos eventos no se perderían gracias a la cola transaccional OpenJMS que tenemos integrada en el sistema.

```
public void ejecutar(){
        try{
          //Lanzar Correlador
          for(int i=0;i<this.getListaCorrelacion().size();i++){
            this.lanzarCorrelador((TareaSW)this.getListaCorrelacion().get(i));
          }
          //Lanzar Recolector
          for(int j=0;j<this.getListaRecoleccion().size();j++){
            this.lanzarRecoleccion((TareaSW)this.getListaRecoleccion().get(j));
          }

          this.estadoEje.setFinish(false);
          this.getEstatusEje().setFailed(false);
        }catch(Exception e){
          log.error("Error al ejecutar la tarea de Monitor: ", e);
          this.getEstatusEje().setFailed(true);
          this.estadoEje.setFinish(true);
        }
}
```

Ilustración 36 Implementación ejecutar Tarea Monitor

La inicialización de la tarea de correlación, sigue el siguiente orden. En primer lugar se obtiene a través de una clase que implementa el patrón factoría, aquel motor de correlación que deseemos, en este caso, un motor de correlación Esper. Una vez conseguida la instancia de motor, se le indica que los eventos que va a procesar tienen un formato Syslog. Obtenemos de la tarea de correlación el mapa de reglas EPL especificadas desde la parte gráfica. Por cada regla dentro de la tarea de correlación, se creará un listener que hemos denominado “Escuchante”. Este listener, se ejecutará de forma asíncrona cuando las reglas se cumplan. Cada regla tiene asociado un listener y un conjunto de alertas, que se añaden como atributo de clase del propio escuchante. Una vez configurado el motor de correlación con los listener, se asigna a la tarea de correlación para que pueda usarla en el método ejecutar. Finalmente la tarea de monitor invoca el método ejecutar de las tareas de correlación que tiene asignadas en su lista y de los recolectores asociados.

3.5.2 Seguridad

La seguridad a cumplir por nuestra aplicación, específica que el servidor Web Apache Tomcat 6.0 deberá ser autenticado a través de un certificado digital, ofreciendo así una autenticación simple por parte única del servidor de aplicaciones. Ante la necesidad de cumplir el requisito, es necesario obtener un certificado digital, este certificado para que tuviera realmente validez, debería ser generado por una autoridad certificadora oficial, como sería la FNMT. En las fases de desarrollo y pruebas de la aplicación, se ha usado certificados de prueba de validez nula, simplemente con el objetivo de probar el correcto funcionamiento del servidor de aplicaciones Tomcat 6.0.

En el momento de configurar el canal seguro de comunicación del servidor de aplicaciones Apache Tomcat 6.0, se realizaron varios pasos previos a este, como son la generación del certificado de prueba, que a continuación se comenta:

- **Generación de un certificado digital de prueba:** Para poder generar el certificado debemos utilizar la herramienta "keytool" que proporciona Java. La utilidad se encuentra en el directorio "bin" de donde se encuentre el JDK. A través del comando "keytool –genkey –alias tomcat –keyalg RSA" y contestemos una serie de preguntas, que serán utilizadas para construir el certificado digital.

- **Localización del fichero de claves keystore:** Las claves certificadoras que acabamos de generar se encontrarán en un directorio denominado "keystore", donde su ubicación pueda variar, aunque habitualmente se encuentra dentro del directorio raíz predefinido por usuario actual.

- **Configuración del servidor de aplicaciones Apache Tomcat 6.0:** Se debe activar un conector para que soporte SSL, modificando el fichero "server.xml", situado en el directorio "conf" del servidor de aplicaciones. Lo habitual es descomentar las líneas que hablen de "SSL HTTP/1.1 Connector on port 8443", de esta forma se activará por defecto el puerto 8443 para SSL. El puerto por defecto para HTTPS es el 443, aunque se recomienda cambiarlo. Una vez configurado el puerto, se deberá especificar la ubicación de las claves, introduciendo en el atributo "keystoreFile" el lugar donde se ubican. Si además al generar el certificado digital, nos pedirá una contraseña que añadiremos al atributo "keystorePass". Es importante recalcar que para que se realice una autenticación simple es necesario que el atributo "clientAuth", contenga el valor "false".

- **Configuración de acceso único a través de https al sistema:** Es necesario impedir el acceso a la aplicación a través de http, por lo que se necesita modificar el fichero "web.xml" del servidor de aplicaciones, indicando los directorios o archivos en los que vamos a requerir una conexión segura. En nuestro caso será toda la aplicación, por lo que dentro de la etiqueta "security-constraint", aparece una etiqueta llamada "url-pattern", la cual hace referencia al patrón que debe seguir para realizar la conexión segura. Para proteger toda la aplicación se rellenará con el patrón "/*". Además en la etiqueta "transport-guarantee", lo rellenaremos con la constante "CONFIDENTIAL", mencionando obviamente nuestro cometido.

Con esta modificación, cualquier acceso a la aplicación y al puerto 8080, será redireccionado al puerto 8443 bajo HTTPS. Las siguientes ilustraciones 37 y 38, mostrarán el estado final de las secciones principales, de los ficheros de configuración, según hemos comentado antes.

```
<security-constraint>
  <display-name>SCGES</display-name>
  <web-resource-collection>
    <web-resource-name>secure</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>

  <user-data-constraint>
    <description>SSL required</description>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

Ilustración 37 Configuración web.xml

```
<Connector SSLEnabled="true" clientAuth="false"
keystoreFile="${user.home}/certificados/keystore/serverKeystore.jks"
keystorePass="123456" maxThreads="150" port="7443"
protocol="HTTP/1.1" scheme="https" secure="true" sslProtocol="TLS"/>
```

Ilustración 38 Configuración conector server.xml

3.6 Pruebas

En éste apartado se describen las pruebas del sistema.

3.6.1 Gestión de roles:

Para poder acceder al sistema, es necesario que un usuario esté registrado previamente en la aplicación y poder así autentificarse correctamente. Para ello, el usuario deberá introducir un nombre de usuario y una contraseña, como muestra la ilustración 39, esperando que el proceso de validación de la información introducida concuerde, con el de la base de datos de usuarios registrados. En el caso que el usuario introduzca incorrectamente sus credenciales, el sistema le informará que la información introducida no es correcta.

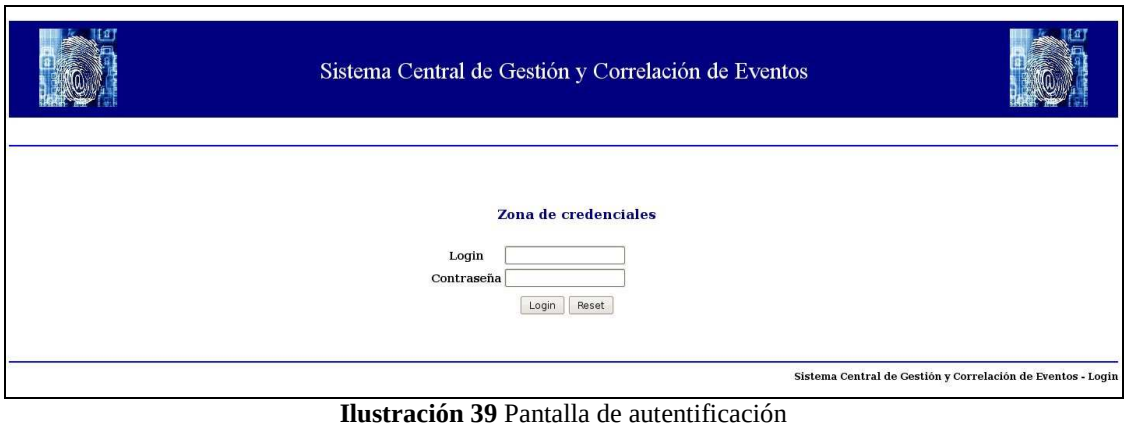


Ilustración 39 Pantalla de autentificación

Una vez autentificado, el usuario podrá comenzar a realizar las operaciones que le tiene permitido sus permisos dentro de la aplicación. La organización visual de la aplicación se divide en dos zonas principales, en el lado izquierdo, se encuentra aquellas acciones que puede realizar el usuario y en el lado derecho, se encuentra su zona de trabajo. Cada usuario de la aplicación puede cambiar su información del perfil dentro del sistema. Entre los datos a cambiar se encuentra su contraseña y su nombre a mostrar, como información no editable se encuentra el nombre de usuario de la aplicación.

El usuario que posea permisos de administración en la gestión de roles, podrá realizar las operaciones de alta, baja, edición, visualización y borrado de usuarios, grupos y políticas. Dentro de las operaciones que puede realizar en la gestión de usuarios, se encuentra, la creación de un nuevo usuario. Para poder dar de alta un nuevo usuario al sistema, el administrador debe seleccionar la opción del menú llamada “Crear Usuario”, una vez presentada la pantalla, debe rellenar los campos del formulario requeridos, entre la información requerida se encuentra, el nombre de usuario, el cual debe ser único dentro del sistema, una contraseña, un nombre a mostrar, grupos a los que el usuario estará asociado. La elección del grupo al que se asociará el nuevo usuario es una decisión crítica, ya que los permisos del usuario dentro de la aplicación, estará relacionados con la política asociada a ese o esos grupos, tal y como se muestra en la ilustración 40.

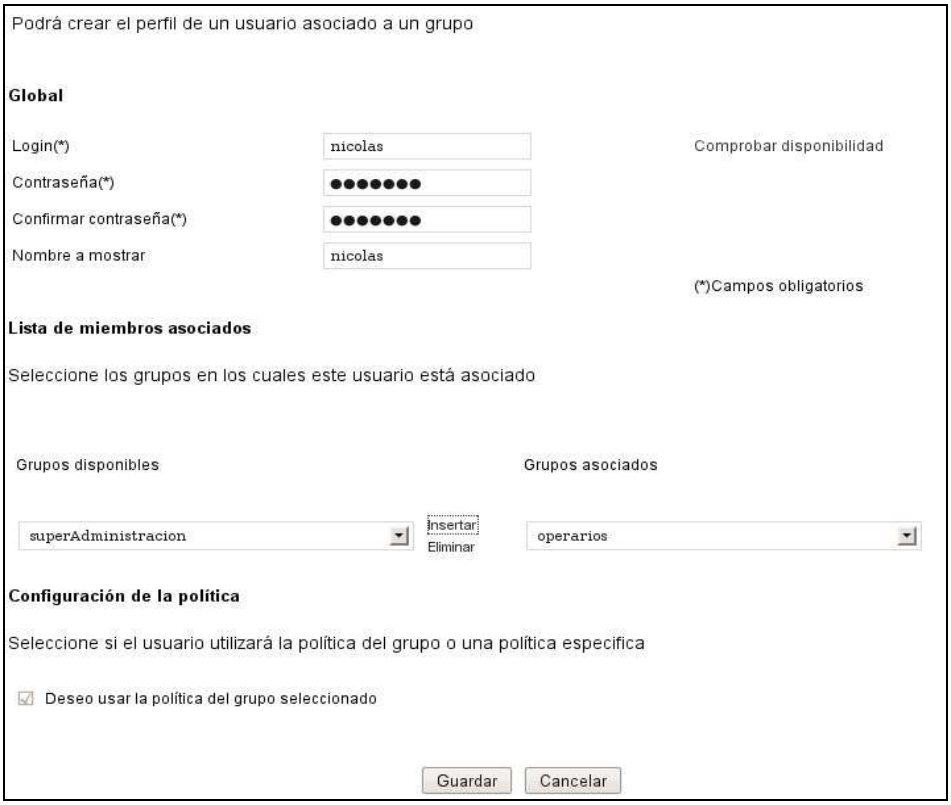


Ilustración 40 Creación de un usuario nuevo

En el apartado de gestión de roles sobre usuarios del sistema, existe la posibilidad de listar todos los usuarios que están registrados en la aplicación, como se observa en la ilustración 41. Esta operación es de gran utilidad, ante la necesidad de obtener información de un usuario registrado. Sólo se podrá visualizar la información de otros usuarios, aquel usuario que posea los permisos de administración de roles dentro de la aplicación. La organización de presentación de usuarios, se gestiona a través de los grupos registrados, excepto en el caso de que se desee ver todos los usuarios registrados de la aplicación, por lo que estarán disponibles en el grupo "All". Además, para realizar operaciones de editar y borrar usuarios registrados. En la edición de un usuario, se podrá editar todos aquellos campos excepto el nombre de usuario. En la operación de borrado, se seleccionará aquel usuario que se desee eliminar y se pulsará el botón "Borrar".

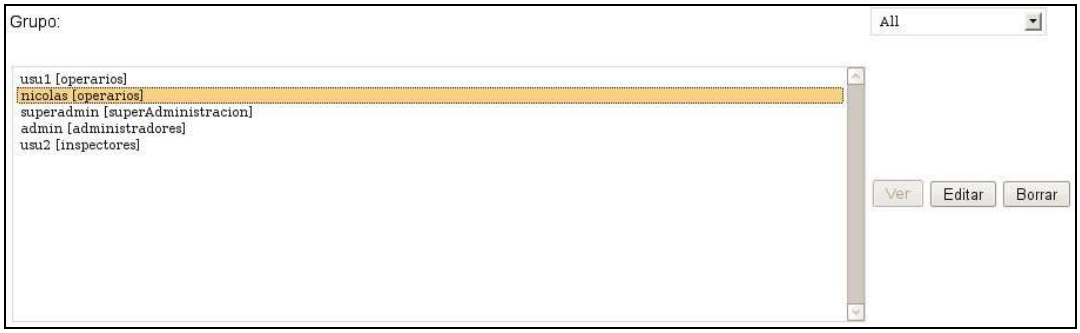


Ilustración 41 Listar todos los usuarios del sistema

En la sección de gestión de roles sobre grupos de la aplicación, existe al igual que las operaciones sobre usuarios, la posibilidad de realizar las operaciones de alta, baja, edición, visualización y borrado de grupos. El usuario que posea permisos de administración de roles, podrá crear un nuevo grupo, introduciendo el nombre del grupo, el cual debe ser único dentro del sistema, el nombre del grupo a mostrar, y un comentario que describa brevemente las propiedades del grupo. Además de los datos básicos requeridos, es necesario que el administrador asocie a los usuarios registrados en el sistema al grupo nuevo creado, para que formen parte de él. Por último, se deberá asociar el nuevo grupo a una política ya configurada en la aplicación, como indica la ilustración 42.

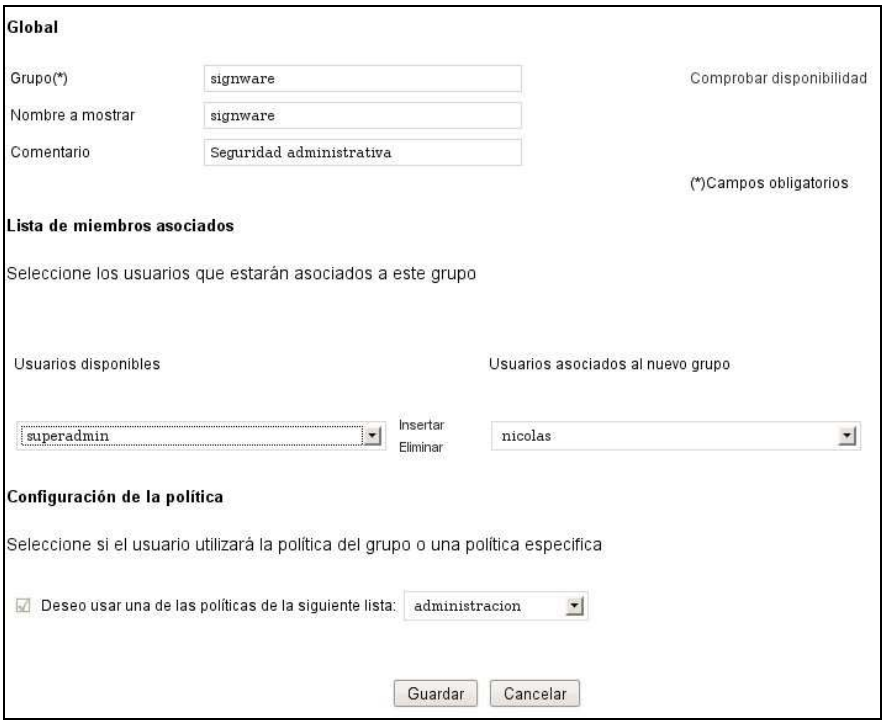


Ilustración 42 Creación de un nuevo grupo

Si el administrador desea realizar la operación de visualización de todos los grupos disponibles del sistema, como se muestra en la ilustración 43, deberá seleccionar la opción "Visualizar Grupo". Una vez seleccionado el grupo, se podrá realizar tres operaciones. La visualización de la información del grupo, la edición de la información, excepto del nombre del grupo y el borrado del grupo seleccionado dentro del sistema. Todos aquellos usuarios que estén únicamente asociados a este grupo, formarán parte del grupo "All".

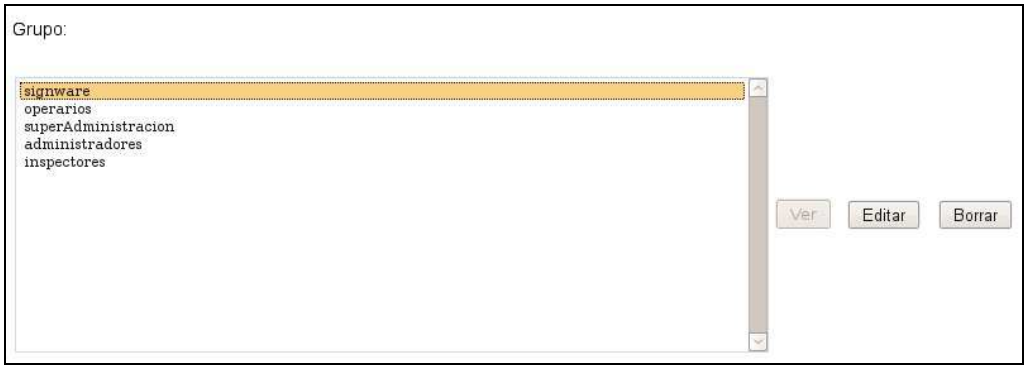


Ilustración 43 Listar todos los grupos del sistema

Al igual que las acciones sobre usuarios y grupos del sistema, es posible realizarlos sobre las políticas. Una política está asociada a los permisos establecidos dentro de la aplicación. Para poder crear una nueva política, el administrador seleccionará la opción “Crear Política”. Como muestra la ilustración 44, información básica a rellanar, se encuentra el nombre único de la política, el nombre a mostrar en la aplicación y un comentario que describa brevemente la finalidad o el objetivo de esa política. Como decisión crítica, en el momento de crear una nueva política, es necesaria la asociación de los permisos disponibles en la aplicación. Cada permiso agrupa un conjunto de operaciones básicas dentro del sistema.

Ilustración 44 Creación de una nueva política

Para poder visualizar, editar o borrar una política, la operación es la misma que en los usuarios y grupos. Seleccionando la opción de “Visualizar Política”, tal y como se muestra en la ilustración 45, se podrá realizar la operación de visualización de información de una determinada política. Si se desea editar, se podrá editar todos aquellos campos excepto el nombre de identificación de la política.

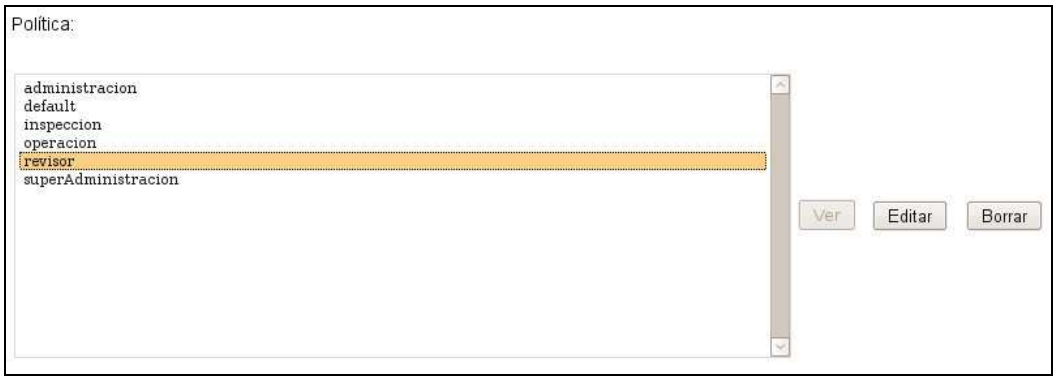
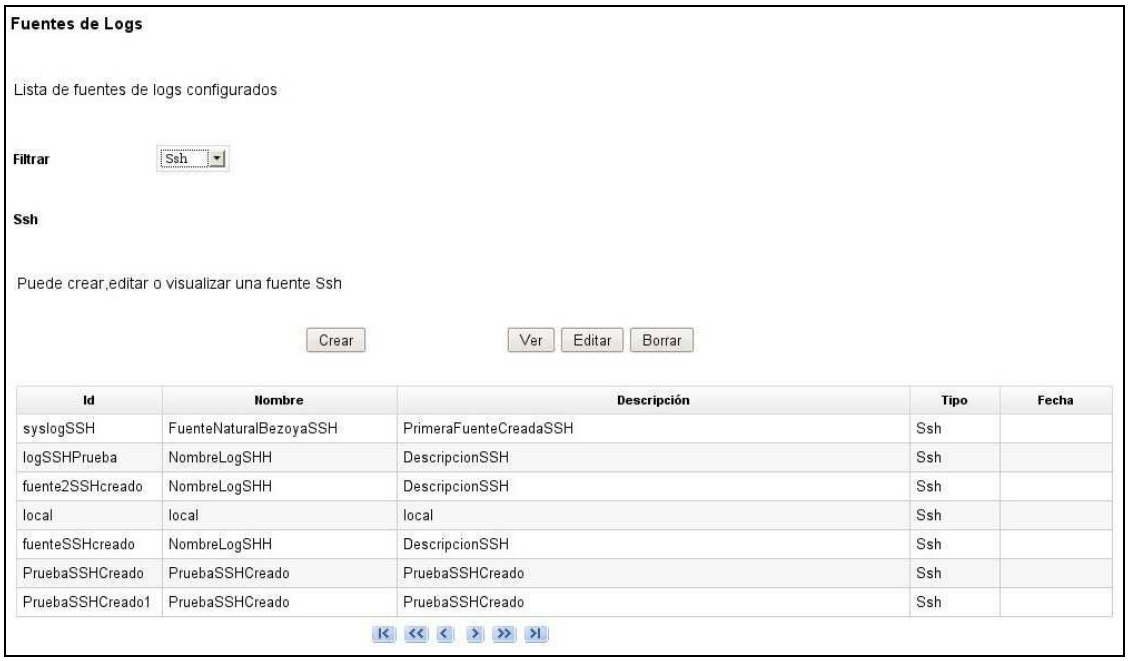


Ilustración 45 Listar todas las políticas del sistema

3.6.2 Gestión de tareas:

A continuación se realizará la configuración y ejecución de las tareas críticas del sistema. Se comentará la configuración de fuentes de eventos, recolección y correlación como tareas básicas, dentro de la monitorización. Para acceder a las operaciones de creación, visualización, edición y borrado de una fuente, es necesario que el usuario con los permisos apropiados seleccione "Fuentes de Logs" sobre el menú. Una vez seleccionado, se deberá elegir sobre el tipo de fuente de evento que se desea configurar. A continuación, se visualizará una tabla con todas las fuentes del tipo seleccionado que ya han sido configuradas con anterioridad. Existe una serie de operaciones que se pueden realizar sobre las fuentes disponibles, estás operaciones se podrán ejecutar siempre que el usuario posea los permisos oportunos, como se muestra en la ilustración 46.



Id	Nombre	Descripción	Tipo	Fecha
syslogSSH	FuenteNaturalBezoyaSSH	PrimeraFuenteCreadaSSH	Ssh	
logSSHPrueba	NombreLogSHH	DescripcionSSH	Ssh	
fuente2SSHcreado	NombreLogSHH	DescripcionSSH	Ssh	
local	local	local	Ssh	
fuenteSSHcreado	NombreLogSHH	DescripcionSSH	Ssh	
PruebaSSHCreado	PruebaSSHCreado	PruebaSSHCreado	Ssh	
PruebaSSHCreado1	PruebaSSHCreado	PruebaSSHCreado	Ssh	

Ilustración 46 Listar todas las fuentes de SSH

En la creación de una fuente depende del tipo de fuente eventos que se haya seleccionado en el filtro. Si se desea crear una fuente de Syslog, como indica la ilustración 47, es necesario rellenar el identificador único de la fuente, un nombre y una descripción. Se puede añadir una descripción más particular sobre la aplicación, máquina y puerto que generan el evento.

Ilustración 47 Creación de una nueva fuente Syslog

En la creación de una fuente SSH cuyo formato de eventos sea de tipo Syslog, es necesario rellenar aquellos campos básicos que requieren un identificador único de fuente, un nombre y una descripción. Además, se puede rellenar una descripción más particular de la máquina, aplicación y puerto, que generan los eventos. En el caso particular de una fuente SSH, la fuente debe estar asociada una cuenta de telnet, que contendrá la información necesaria para establecer una conexión SSH con la máquina remota. Es necesario indicar, qué fuente principal y rotada deben recoger los eventos. Esta opción puede realizar una búsqueda de la fuente remota con la máquina donde se generen los eventos, tal como se observa en la ilustración 48.

Ilustración 48 Configuración específica de una fuente SSH

Para poder configurar una tarea de recolección, se debe seleccionar en el menú de la aplicación "Recolección". Dentro de los tipos de recolectores, seleccionar el tipo de recolector que se desea configurar. Si seleccionamos filtrar por recolectores de tipo Syslog, podremos visualizar todos los recolectores syslog configurados en el sistema. Sobre la lista de recolectores configurados, se podrá realizar las operaciones de creación, edición, visualización y borrado, si se tiene los permisos oportunos. En la creación de un recolector de tipo Syslog, es necesario introducir un identificador único de recolección, un nombre y una descripción breve. Como sección particular del recolector, se debe introducir la dirección IP y el puerto donde se ejecutará. Una vez configurados estos datos, representados en la ilustración 49, se seleccionarán aquellas fuentes de eventos de tipo Syslog, que ya han sido configuradas con anterioridad.

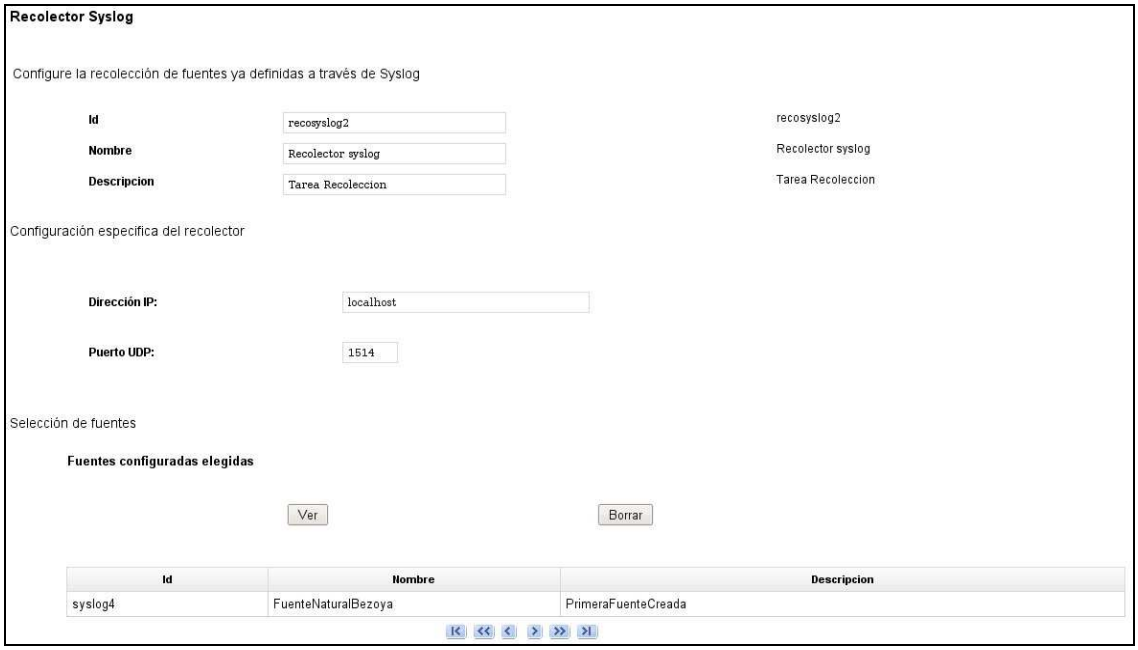


Ilustración 49 Creación de un recolector Syslog

En la creación de un recolector de tipo SSH, el proceso es prácticamente el mismo procedimiento que un recolector tipo Syslog. Se introducirá un identificador único de fuente, un nombre y una breve descripción. En su configuración particular de recolector, se rellenará el tiempo de intervalo de consulta con la fuente y finalmente, se realizará la selección de las fuentes de SSH ya configuradas asociadas al recolector, tal y como se presenta en la ilustración 50.

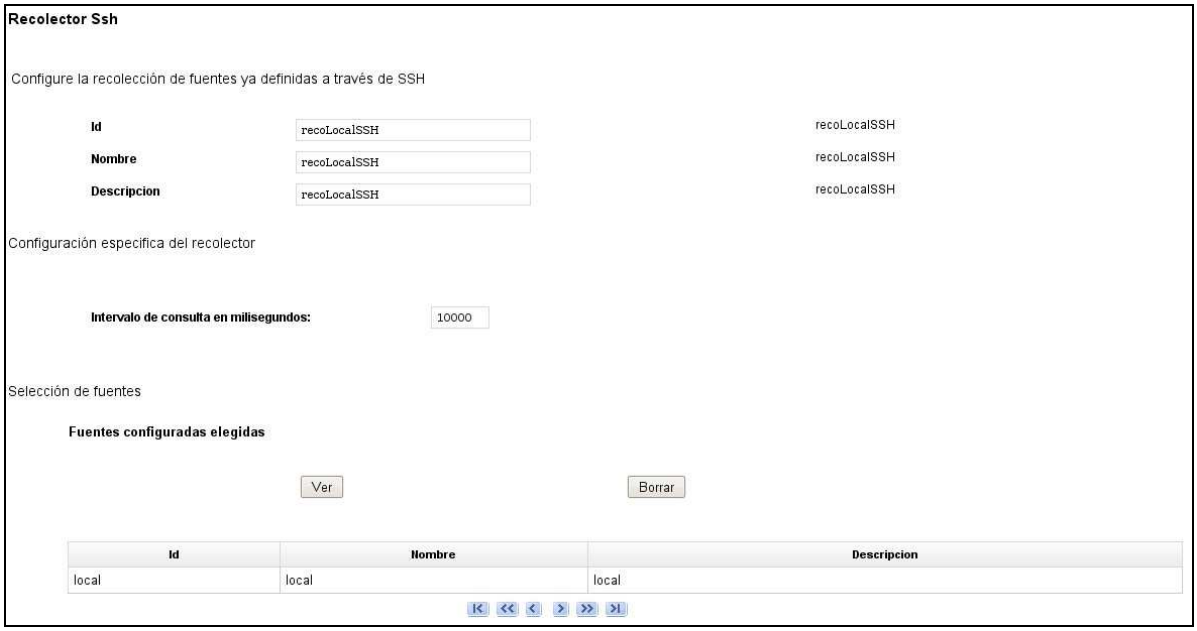


Ilustración 50 Creación de un nuevo recolector SSH

Para poder configurar una tarea de correlación, es necesario seleccionar en el menú “Correlación”. Se podrá visualizar la lista de tareas de correlación configuradas en el sistema y sobre los que se puede realizar las operaciones de ver, editar y borrar. Para crear una tarea de correlación, se introduce un identificador único de tarea de correlación, un nombre y una breve descripción. Una vez introducidos los datos obligatorios, se pueden crear las reglas de correlación, como se indica en la ilustración 51.

Ilustración 51 Creación de una nueva tarea de Correlación

Como se puede observar en la ilustración 52, una regla de correlación está formada por un identificador único de regla, y una regla en lenguaje EPL. En la selección del tipo de alarma, existe la posibilidad de elegir alarmas predefinidas, contextuales o ambas. Para configurar una alarma predefinida, se debe seleccionar el tipo de alarma, el destinatario de la alarma y el mensaje contextualizado de la alarma. En el caso de que se desee seleccionar una tarea ya configurada como alarma, simplemente se selecciona la tarea de la lista e insertarla en la lista de alarmas asociadas a la regla de correlación.

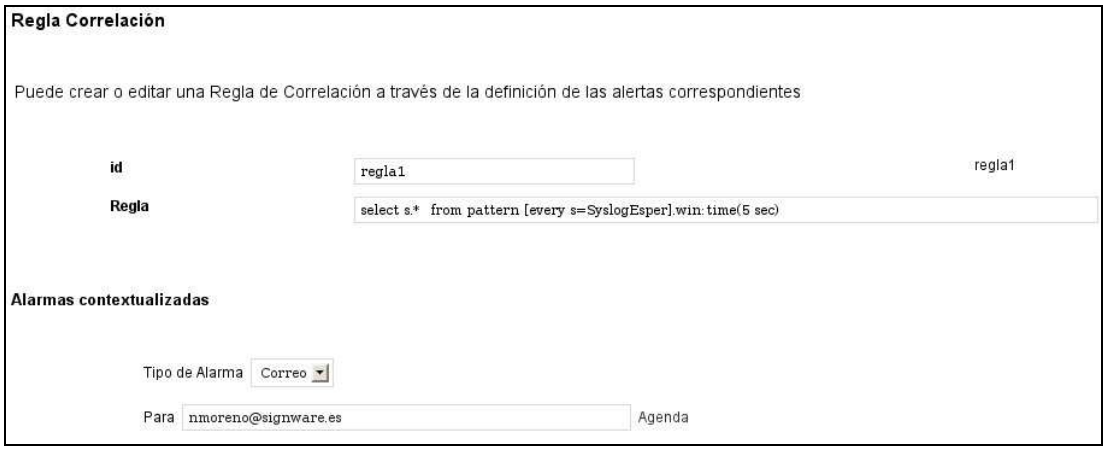


Ilustración 52 Creación de una nueva regla de correlación

Si se desea el sistema da la posibilidad de crear nuevos mensajes contextuales para las alarmas de la reglas de correlación, como se observa en la ilustración 53.

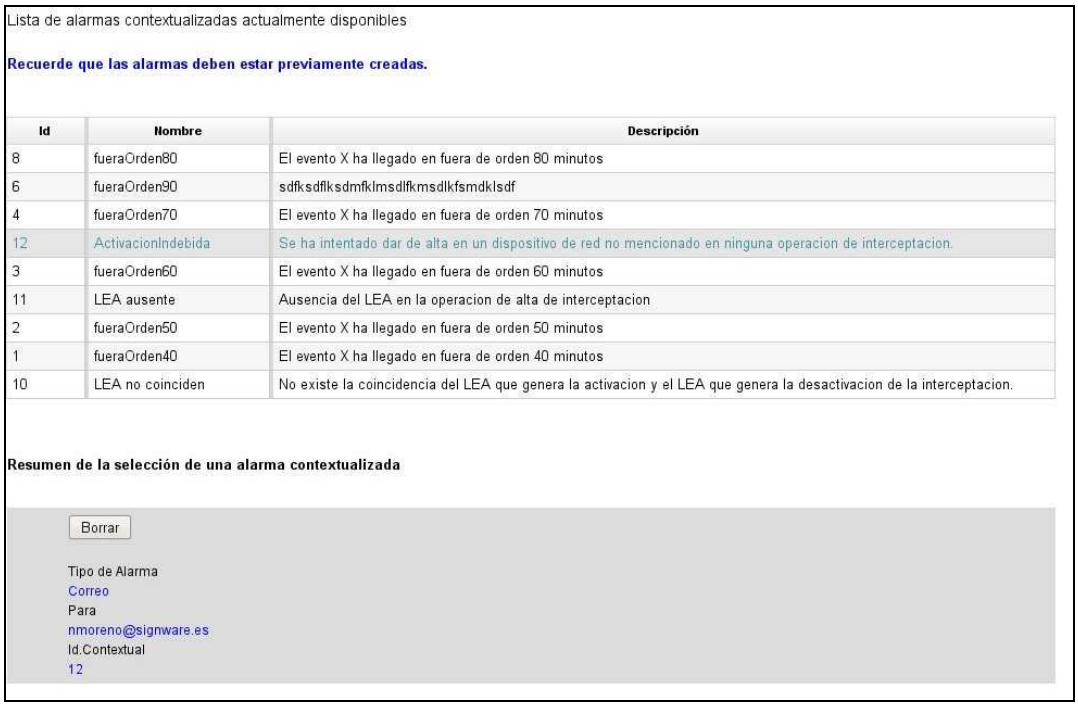


Ilustración 53 Selección de alarmas contextualizadas

En la configuración de una tarea de monitor, el usuario de la aplicación debe pulsar sobre la opción de menú "Monitor", donde se le presentará una lista de las tareas de monitor que ya han sido configuradas. Sobre las tareas de monitor configuradas, se podrán realizar las operaciones de ejecución, visualización, edición y borrado, si se posee los permisos oportunos. La creación de una tarea de monitorización, se ha definido en una serie de configuraciones parciales a modo de asistente. En el primer paso, el usuario debe introducir un identificador único de tarea de monitor, un nombre y una descripción, como representa la ilustración 54.

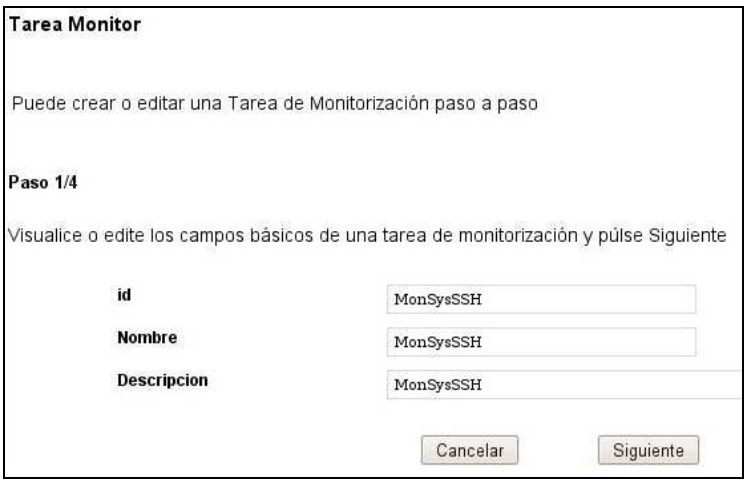


Ilustración 54 Creación de tarea de monitor Paso 1

En el segundo paso, se realizará la selección de todas las tareas de recolección disponibles configuradas que se asocian al monitor, reflejado en la ilustración 55.

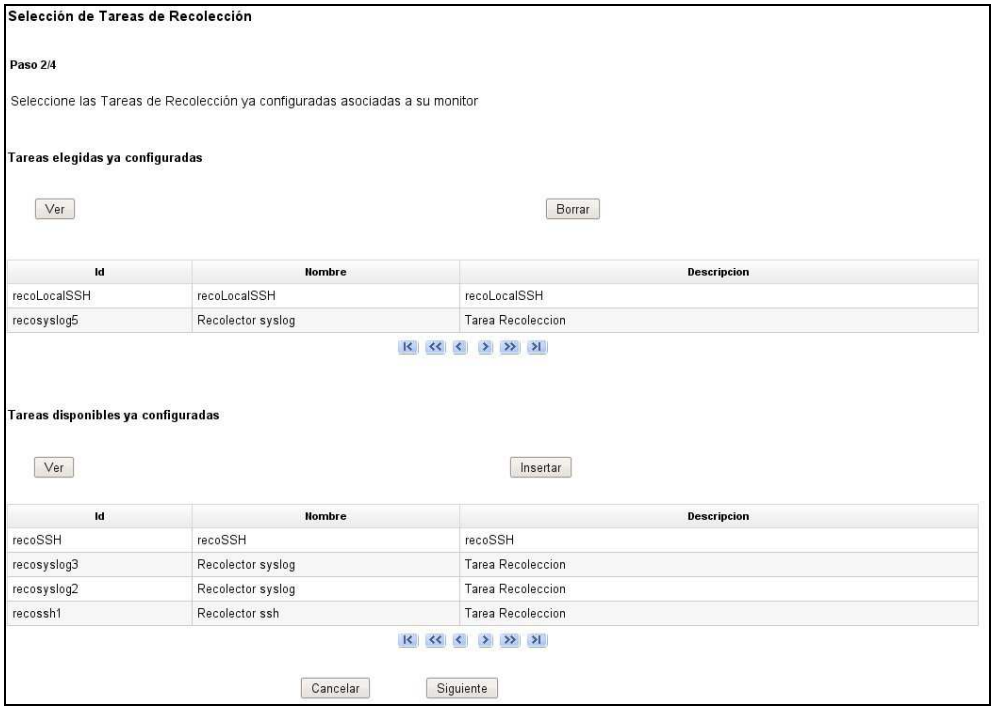


Ilustración 55 Creación de tarea de monitor Paso 2

El tercer paso, como se puede observar en la ilustración 56, se seleccionan las tareas de correlación en relación con los recolectores elegidos.

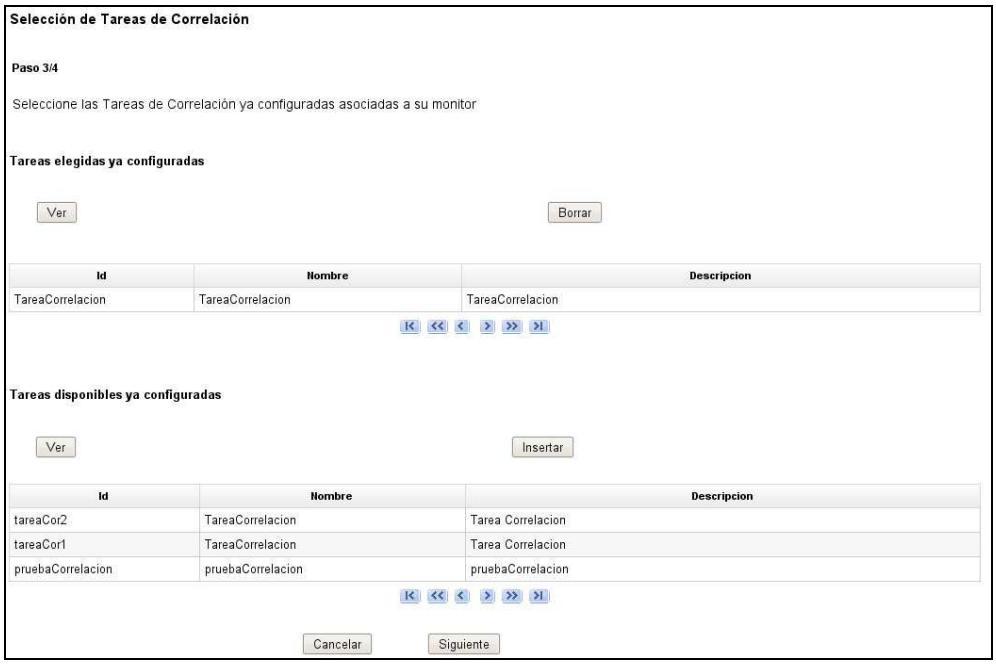


Ilustración 56 Creación de tarea de monitor Paso 3

Finalmente se genera un resumen de las configuraciones, a través de un diagrama que se muestra en la ilustración 57, que representa los recolectores seleccionados asociados al monitor y las fuentes respectivas.

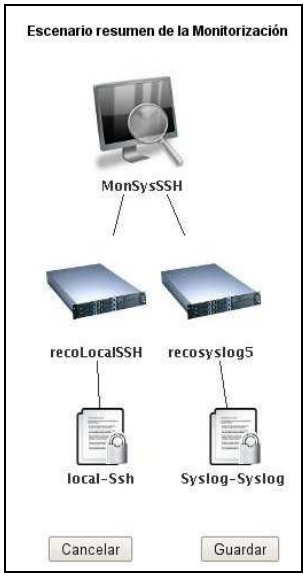


Ilustración 57 Resumen tarea monitor Paso 4

Una vez configuradas las fuentes de eventos, los recolectores, los correladores y la tarea de monitor, se ejecutará el monitor que pondrá en marcha cada uno de sus subtareas que lo forman. En la ilustración 58, observamos el mensaje de que la ejecución de monitor se ha realizado correctamente.

Monitores disponibles

Lista de tareas de monitorización ya configuradas

- Se ha ejecutado CORRECTAMENTE la tarea: MonSysSSH

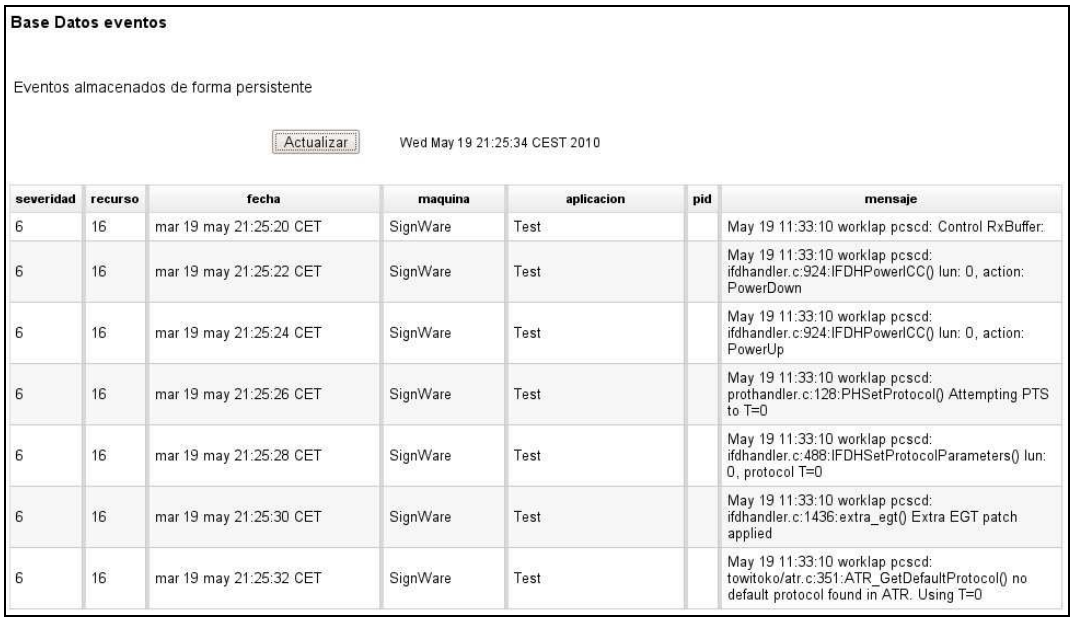
Nueva Tarea | Ver | Editar | Borrar | Parar | Ejecutar

Id	Nombre	Descripción	Planificación	Estado	Fecha
MonSysSSH	MonSysSSH	MonSysSSH		Ejecutando	
MonitorSSH	MonitorSSH	MonitorSSH		Parado	
tareaMonitor	TareaMonitor	Tarea Monitor		Parado	

Ilustración 58 Confirmación de ejecución correcta de la Tarea Monitor

3.6.3 Visualización de eventos

Todos los eventos que se reciben al sistema central de gestión y correlación de eventos son almacenados de manera persistente a través a una base de datos HSQL. Como se puede observar en la Ilustración 59, los eventos son parseados, es decir, estructurados y diferenciados, con el fin de poder realizar una posterior correlación.



Base Datos eventos

Eventos almacenados de forma persistente

Actualizar Wed May 19 21:25:34 CEST 2010

severidad	recurso	fecha	maquina	aplicacion	pid	mensaje
6	16	mar 19 may 21:25:20 CET	SignWare	Test		May 19 11:33:10 worklap pcscd: Control RxBuffer:
6	16	mar 19 may 21:25:22 CET	SignWare	Test		May 19 11:33:10 worklap pcscd: ifdhandler.c:924:IFDHPowerICC() lun: 0, action: PowerDown
6	16	mar 19 may 21:25:24 CET	SignWare	Test		May 19 11:33:10 worklap pcscd: ifdhandler.c:924:IFDHPowerICC() lun: 0, action: PowerUp
6	16	mar 19 may 21:25:26 CET	SignWare	Test		May 19 11:33:10 worklap pcscd: prothandler.c:128:PHSetProtocol() Attempting PTS to T=0
6	16	mar 19 may 21:25:28 CET	SignWare	Test		May 19 11:33:10 worklap pcscd: ifdhandler.c:488:IFDHSetProtocolParameters() lun: 0, protocol T=0
6	16	mar 19 may 21:25:30 CET	SignWare	Test		May 19 11:33:10 worklap pcscd: ifdhandler.c:1436:extra_egt() Extra EGT patch applied
6	16	mar 19 may 21:25:32 CET	SignWare	Test		May 19 11:33:10 worklap pcscd: towitoko/atr.c:351:ATR_GetDefaultProtocol() no default protocol found in ATR. Using T=0

Ilustración 59 Visualización eventos recibidos

La visualización de los eventos recibidos es de gran utilidad en el proceso de inspección y análisis forense. La necesidad de estructurar la información recibida del evento y transformarlo en un objeto útil, requiere un enorme trabajo de parseo de la información.

Capítulo 4
Conclusiones

Una vez finalizado el proceso de desarrollo de la aplicación, se ha obtenido un sistema que satisface los objetivos marcados en la especificación de requisitos. La aplicación se ha construido de forma eficaz, funcional y modular, con el fin de facilitar su evolución y mantenimiento en el futuro, características que son necesarias y casi obligatorias en el mundo de los sistemas de seguridad. Las pruebas realizadas han resultado en su totalidad satisfactorias, el sistema cumple con sus objetivos, ayudando de forma activa a los técnicos de seguridad de cualquier organización que necesite optimizar sus mecanismos de seguridad, evitando momentos de incertidumbre y pérdida de tiempo en el análisis. Las principales conclusiones que hemos obtenido son las siguientes:

- **Necesidad de establecer medidas de seguridad**: La necesidad de realizar un estudio de las vulnerabilidades que están expuestos nuestros sistemas, puede garantizar un nivel de protección adecuada de la información a salvaguardar. El hecho de ser precavidos, puede ahorrarnos disgustos y sorpresas en el futuro.

- **Tecnologías que ayudan a proteger nuestro sistema**: El uso de las diferentes herramientas que ayudan a prevenir las posibles vulnerabilidades del sistema, es una buena opción como primera barrera de protección frente a ataques, independientemente del uso de un sistema centralizado de gestión y correlación de eventos.

- **Uso de los registros de los dispositivos**: El registro de las operaciones realizadas en el sistema por los propios usuarios y aplicaciones, es una información muy valiosa que debe ser frecuentemente consultada. Lo común es hacer referencia a él, cuando ocurre algún problema o imprevisto.

- **Tratamiento y selección de la información**: Toda la información que registra el sistema no es del todo útil. Es necesario saber diferenciar que es lo realmente importante, pudiendo así realizar un estudio y reconstrucción detallada del suceso ocurrido.

- **Necesidad de avisos en tiempo real**: El tiempo de respuesta es prioritario en sistemas críticos, ya que una rápida respuesta puede atajar un daño mayor.

Asimismo, hemos encontrado las siguientes dificultades:

- **Estudio del dominio:** La necesidad de comprender con detalle el problema que se desea solucionar, obliga a entender los conceptos propios del negocio. El dominio técnico de la seguridad es complejo, esto es debido al ingenio y la astucia de los agentes dañinos. Es necesario familiarizarse con el contexto del problema, entendiendo porqué surgen las vulnerabilidades en el sistema y estudiar la solución más acorde, según las necesidades.

- **Propuesta de una solución y selección de la tecnología adecuada:** La propuesta desarrollo que resuelva las vulnerabilidades así como la elección de la tecnología que resuelva los problemas planteados.

- **Dificultades en el análisis, diseño e implementación de la aplicación:** El hecho de implementar una herramienta de seguridad, me exigía un análisis y diseño detallado y preciso. El uso de diagramas UML para la etapa de diseño ha sido fundamental. El uso de patrones de diseño en la etapa de diseño e implementación, me ha ayudado a resolver dificultades, que se presentaba en cada componente.

Como futuras mejoras proponemos las siguientes:

- **Ampliar la compatibilidad con mayor número de fuentes de eventos**: La necesidad real de poder aceptar y procesar mayor número de tipos de formatos libres y propietarios, es una cuestión a mejorar. Para ello es necesario proponer un formato común de campos con el fin de simplificar el procesamiento de eventos.

- **Facilitar la generación de reglas de correlación:** La tarea de definición y especificación de reglas que se ajusten a las necesidades de las organizaciones, con el fin de protegerse de las posibles amenazas, es a menudo complejo. Estas acciones son realizadas por profesionales técnicos expertos en el lenguaje. Facilitar a través de un asistente gráfico de fácil manejo, en la creación y especificación de reglas de correlación, sería muy interesante.

- **Aumentar el número de tipos de alertas:** El envío de mensaje de correo como acción pasiva ante una vulnerabilidad, es una solución aceptable. La posibilidad de ampliar a mensajes de texto a un dispositivo móvil, a través de dispositivos GPRS, integrados en el sistema.

- **Generación de gráficas, informes e integración de motores de búsqueda:** El uso de procesos de análisis forense, con el fin de descubrir e investigar la acción que provocó el daño, es muy importante. El poder generar informes, gráficas representativas del estado del sistema y obtener la información necesaria a través de búsquedas, son operaciones de gran utilidad.

- **Aumentar el nivel de seguridad del sistema:** El envío de forma segura de los eventos a través de syslog y la posibilidad de garantizar la integridad de los eventos recibidos, impidiendo su manipulación, serían mejoras muy interesantes.

Bibliografía

[AYL 08] Víctor Ayllón, Juan M.Reina "CEP/ESP: Procesamiento y correlación de gran cantidad de eventos en arquitecturas SOA"
Artículo publicado en www.novayre.es en 2008

[BAR 05] Alejandro Barrera García- Orea "Presente y futuro de los IDS"
Artículo publicado en Whitepapers de neurosecurity.com en 2005

[BOO 00] Grady Booch, James Rumbaugh, Ivar Jacobson. "El lenguaje unificado de modelado".
Editorial Addison Wesley. Año 2000.

[ECK 09] Michael Eckert, François Bry "Complex Event Processing (CEP)"
Artículo publicado por la Institut fur Informatik, Ludwig-Maximilians-Universitat manchen 2009

[GAL 03] Pablo Galdámez "Seguridad Informática".
Artículo Actualidad TIC 2003 Revista del Instituto Tecnológico de Informática

[GON 03] Diego González Gómez "Sistema de detección de intrusiones"
Documento publicado en www.dgonzalez.net en 2003

[KRU 05] Christopher Krugel, Fredrik Valuer, Giovanni Vigna
Artículo llamado "Intrusión Detection and Correlation" 2005

[LUC 05] D. Luckham: "The Power of Events"
Addison Wesley 2005

[LUC 08] D. Luckham, Roy Schulte "Event Processing Glosary – Version 1.1 2008"
Artículo de conceptos y definiciones para Event Processing Technical Society

[RUI 03] Pablo Ruiz Garcia "Gestión de eventos de seguridad"
Artículo publicado en "Digital Security Research"

[SAL 05] Raúl Salinas "Sistemas de detección de intrusos"
Artículo Actualidad TIC 2005 Revista del Instituto Tecnológico de Informática

[SHE 09] Jerry Sheik "Evaluación de Logger ArcSight"
Artículo patrocinado por la empresa ArcSight en 2009

[OSO 06] Diego Osorio Reina "Correlación de alertas y eventos en seguridad informática"
Artículo publicado en Bogotá 2006

[VER 04] Gabriel Verdejo Alvarez "Seguridad en redes IP"
Artículo Investigación DEA 2004

[VIL 03] Alicia Vila, Máximo Sedano, Ana López, Ángle A.Juan
Árticulo "Correlación lineal y análisis de regresión" publicado en www.uoc.edu

REFERENCIAS ELECTRÓNICAS

[WEB01] Clasificación y Tipos de Ataques Contra Sistemas de Información 2009
Página informativa sobre la seguridad informática de forma genérica.
http://www.canal-ayuda.org/a-seguridad/tipataques.htm

[WEB02] Instituto Nacional de Tecnologías de la Comunicación. 2009
http://cert.inteco.es

[WEB03] Wikipedia Enciclopedia libre 2009
Información a cerca de sistema de detección de intrusiones
http://es.wikipedia.org/wiki/Sistema_de_detecci%C3%B3n_de_intrusos

[WEB04] Wikipedia Enciclopedia libre 2009
Información a cerca de registros de sistemas
http://es.wikipedia.org/wiki/Log_(registro)

[WEB05] Wikilearning 2009
Control, administración e integridad de logs
http://www.wikilearning.com

[WEB06] Wikipedia Enciclopedia libre 2009
Información a cerca de Syslog
http://en.wikipedia.org/wiki/Syslog

[WEB07] Wikipedia Enciclopedia libre 2009
Información a cerca Event-Driven Architecture
http://en.wikipedia.org/wiki/Event-driven_architecture

[WEB08] Event-Driven Architecture Overview 2009
http://elementallinks.typepad.com/bmichelson/2006/02/eventdriven_arc.html

[WEB09] Wikipedia Enciclopedia libre 2009
Información Event Stream Processing
http://en.wikipedia.org/wiki/Event_stream_processing

[WEB10] Event Stream Processing – A new physics of software 2009
http://www.information-management.com/infodirect/20050729/1033537-1.html

[WEB11] Wikipedia Enciclopedia libre 2009
Información a cerca Complex Event Processing
http://en.wikipedia.org/wiki/Complex_event_processing

[WEB12] David Luckham
"What´s the difference betweem ESP and CEP"
http://complexevents.com/2006/08/01/what%E2%80%99s-the-difference-between-esp-and-cep/

[WEB13] Página oficial de Esper Tech Inc.
http://esper.codehaus.org/

[WEB14] Wikipedia Enciclopedia libre 2009
Información plataforma Java
http://es.wikipedia.org/wiki/Plataforma_Java

[WEB15] Ciberaula Java
Lenguaje Java y plataforma J2EE
http://java.ciberaula.com/articulo/que_es_java/

[WEB16] Enciclopedia
Información a cerca de J2EE
http://enciclopedia.us.es/index.php/Java_2_Enterprise_Edition

[WEB17] OpenJMS
Web oficial del servicio Open Source JMS
http://openjms.sourceforge.net/

[WEB18] ICEfaces
Web oficial del framework que combina JSF con AJAX y JavaScript para J2EE
http://www.icefaces.org

[WEB19] Apache Tomcat
Web oficial del servidor de aplicaciones Open Source
http://tomcat.apache.org/

[WEB20] SSL
Web informativa de la seguridad en entornos web.
http://www.iec.csic.es/criptonomicon/ssl.html

[WEB21] Wikipedia Enciclopedia libre 2009
Información sobre los protocolos SSL y TLS
http://es.wikipedia.org/wiki/Transport_Layer_Security

[WEB21] Wikipedia Enciclopedia libre 2009
Información sobre los protocolos SSH y SCP
http://es.wikipedia.org/wiki/Secure_Shell

[WEB22] Wikipedia Enciclopedia libre 2009
Información sobre términos comunes en la seguridad informática
http://es.wikipedia.org/wiki/Seguridad_informática

ANEXO I: HARDWARE Y SOFTWARE UTILIZADOS

Hardware utilizado

Para el desarrollo de la aplicación se ha utilizado un procesador Intel Core 2 Duo P8400 ambos a 2.26 GHz, con una memoria caché L2 de 3MB y un FSB de 800 MHz. Se ha usado una memoria RAM DDR2 de 2.8 GiB, necesaria para poder procesar los eventos recibidos con cierta velocidad. Respecto al almacenamiento secundario, se ha utilizado un disco duro SATA de 320 GB a 5.400 rpm. Es recomendable tener un disco duro de gran capacidad para poder almacenar los eventos recibidos, el tiempo requerido. La tarjeta de red Gigabit Ethernet LAN con una velocidad de 10/100/1000BASE-T. El sistema operativo utilizado es GNU/Linux Ubuntu versión 8.10 Intrepid, con un núcleo de genérico 2.6.27-7.

Software utilizado

De las tecnologías software utilizadas en el desarrollo de la aplicación, y al utilizar la plataforma de desarrollo J2EE y varios frameworks, se ha decido utilizar las siguientes herramientas:

- **Eclipse:** Es un entorno de desarrollo integrado de código abierto multiplataforma, usada habitualmente para desarrollar entornos de desarrollo integrados IDE, idóneo para trabajar con la plataforma J2EE y sus frameworks. Además facilita la integración de servidores de aplicaciones, como es Tomcat 6.0 y su generación de paquetes WAR, para exportar la aplicación a cualquier otro servidor de aplicaciones tipo Tomcat.

- **Syslogd:** Es un demonio de syslog que se lanza automáticamente al arrancar un sistema Unix, es el encargado de guardar informes sobre el funcionamiento de la máquina. Puede recibir mensajes de diferentes partes del sistema y los envía o almacena en diferentes localizaciones, tanto locales como remotas, en nuestro caso, se enviarán a la aplicación de gestión y correlación de eventos de seguridad.

- **Less:** Es un visualizador de archivos de texto de código abierto, que se ejecuta a través del intérprete de comandos. A diferencia de otros comandos similares, “less” permite una completa navegación por el contenido del fichero, utilizando un mínimo de recursos del sistema. Se utilizará a través del protocolo “SSH”, con el fin de obtener los eventos de escritos en un fichero que representa una fuente a monitorizar.

Tecnologías JAVA para el desarrollo de la aplicación

El grueso de la aplicación está desarrollada en el lenguaje de programación Java, además de la incorporación de múltiples herramientas del entorno J2EE o JavaEE. La plataforma Java es originaria de Sun Microsystems y es capaz de ejecutar aplicaciones desarrolladas en lenguaje de programación Java, además de otros lenguajes que compilen a bytecode **[WEB14]**.

Bytecode es el código intermedio más abstracto que el propio código máquina, es tratado como un fichero binario, conteniendo el programa ejecutable. Los programas en bytecode suelen ser interpretados por un intérprete de bytecode, en nuestro caso es la máquina virtual de Java. Su principal ventaja es la portabilidad del código binario, ya que este puede ser ejecutado en diferentes plataformas y arquitecturas.

La máquina virtual de Java (JVM) es un programa en nativo, que se ejecuta en una plataforma específica, capaz de reconocer instrucciones expresadas en Java bytecode. La plataforma Java y las tecnologías que giran entorno a la máquina virtual. Un programa realizado en la plataforma Java necesita además de la máquina virtual, necesita un conjunto de librerías que proporcionan una serie de servicios, cuyo principal objetivo es ayudar a las necesidades de la aplicación. La unión entre estos componentes, se denomina “Java Runtime Environment” o JRE.

Las librerías que proporciona la plataforma Java, ofrecen al programador un conjunto de herramientas comunes, que le son básicas. Además se ofrecen una interfaz abstracta para tareas que son dependientes del hardware y del sistema operativo **[WEB15]**.

Java 2 Enterprise Edition (J2EE)

Es parte de la plataforma Java para desarrollar y ejecutar aplicaciones, con arquitecturas distribuidas, apoyándose en componentes modulares ejecutándose sobre un servidor de aplicaciones, siendo en nuestro caso Apache Tomcat. Java2EE incluye varias interfaces de programación de aplicaciones (API), que define y coordina, esta es la características principal de J2EE, ya que permite desarrollar una aplicación empresarial portable, escalable y flexible con una gran variedad de componentes añadidos.

Una de las principales ventajas como plataforma es empezar sin ningún coste, descargar el entorno de forma gratuita, con innumerables herramientas de código abierto **[WEB16].** En la ilustración 60, podemos observar la arquitectura básica de la plataforma JAVA.

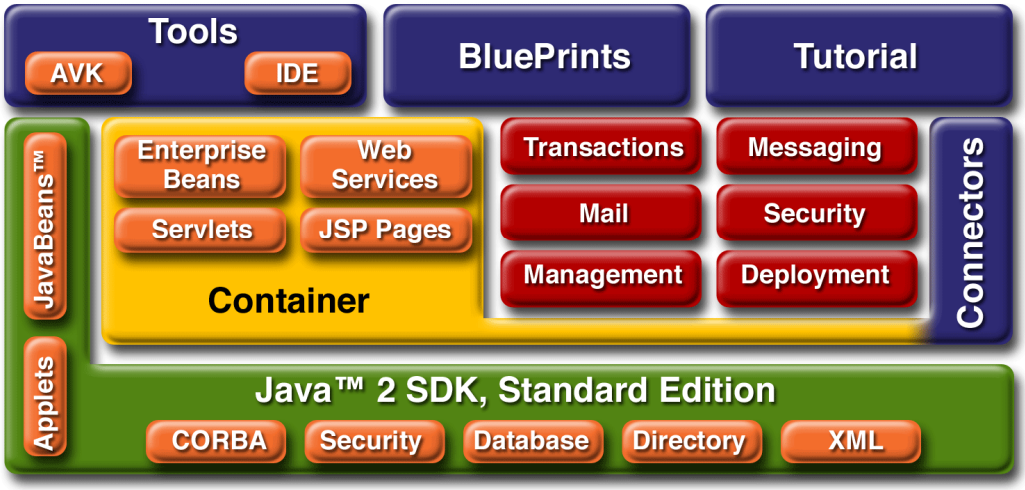


Ilustración 60 Arquitectura básica de la plataforma Java de Sun Microsystems

Open Java Message Service

OpenJMS es un de los principales proveedores Open Source de la API de Java Message Service de Sun Microsystems. Java Message Service es un mecanismo de comunicación asíncrona en sistemas distribuidos, donde su principal utilidad es en el uso de componentes distribuidos débilmente acoplados.

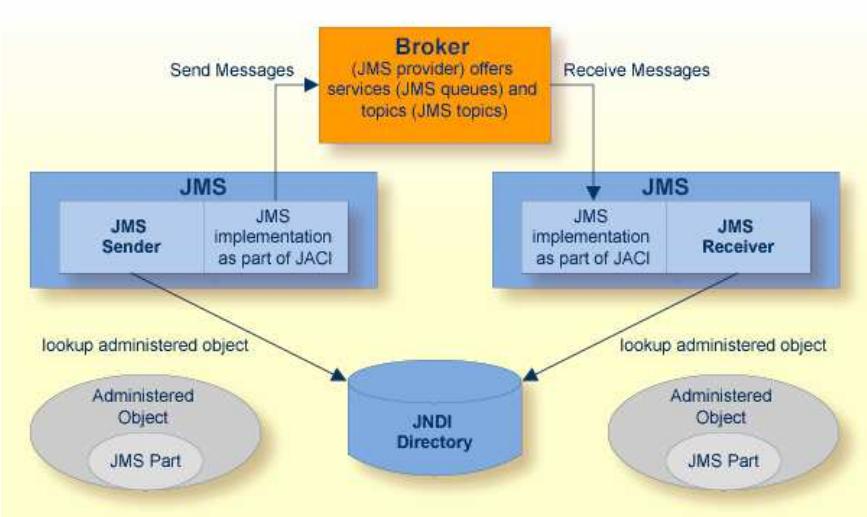


Ilustración 61 Explicación del funcionamiento de JMS

La principal ventaja es que no depende de ningún servidor de aplicaciones y por lo tanto los clientes de diferentes aplicaciones pueden ponerse de acuerdo para la interoperabilidad de sus capas.

OpenJMS incorpora conexiones para diferentes bases de datos, por defecto utiliza Derby. Como se puede observar en la ilustración 61, OpenJMS posee múltiples

usos, como son las colas sincronías y asíncronas, que garantizan que los elementos que se insertan en la estructura no se pierden **[WEB17]**.

ICEfaces

Es un framework de código abierto desarrollado para construir aplicaciones web con AJAX tipo "Rich Internet Application" o RIA. Es administrado por ICEsoft Technologies Inc.

Permite al desarrollador incluir una serie de etiquetas "AJAX-tags" en sus Java Server Pages (JSP) o en sus JSPX (JSP + XML), de tal forma que el código AJAX es generado automáticamente por el propio framework.

ICEfaces abstrae al desarrollador de la aplicación de la tecnología AJAX, ya que no necesita etiquetas especiales. ICEFaces se encarga de enviar sólo la información necesaria entre el cliente y el servidor.

Las aplicaciones desarrolladas con esta tecnología no necesitan plugins de navegador o applets para ser visionadas.

Este tipo de aplicaciones están basadas en la tecnología Java Server Faces (JSF), que permite el desarrollo de aplicaciones JavaEE de forma sencilla. Además es ideal para aplicaciones que tienen cierta dependencia de JavaScript, ya que ICEfaces es muy flexible en ese sentido.

Citar algunas ventajas de esta tecnología:

- Está basado en código abierto.
- Basado en estándares, multitud de plugins para varias IDE de Java.
- AJAX es transparente para el programador.
- Compatibilidad con todos los servidores de aplicaciones.
- Seguridad, compatible con SSL.
- Escalabilidad y clustering.
- Carga las páginas de forma incremental con edición de sección.
- Preserva el contexto del usuario durante la actualización de la página.
- Especialmente diseñado para aplicaciones en tiempo real, recargando las páginas de forma asíncronas.
- Compatibilidad con la totalidad de navegadores comerciales.
- Suite de componentes ya desarrollados **[WEB18]**.

En la ilustración 62, se muestra las diferentes tecnologías que utiliza ICEFaces, junto con las relaciones y comunicaciones que establecen entre ellos.

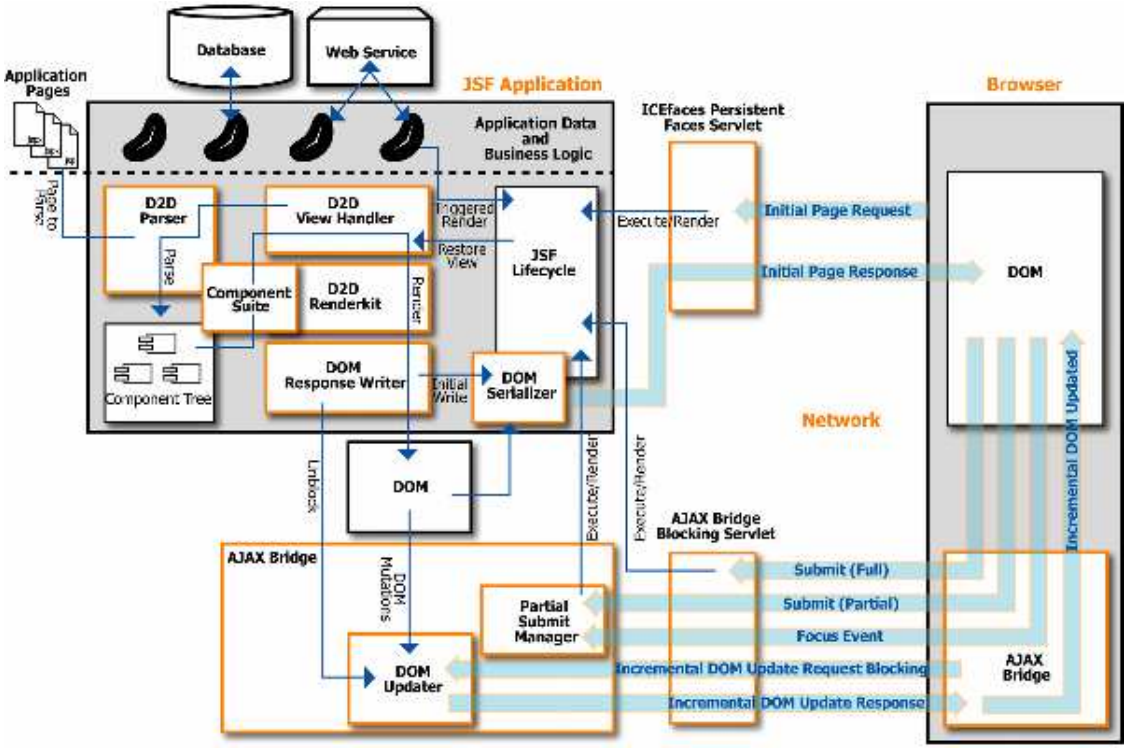


Ilustración 62 Arquitectura básica de ICEfaces

Apache Tomcat

Jakarta Tomcat o Apache Tomcat funcionan como un contenedor de servlets Open Source, desarrollado por el proyecto Yakarta en la fundación "Apache Software Fundation" (ASF). Tomcat está totalmente escrito en el lenguaje de programación Java, por lo que es necesario tener instalado la JVM en el sistema operativo.

Tomcat es un servidor web con soporte de JSPs y servlet de Sun Microsystems. Incluye el compilador Jasper, que compila JSP, dando como resultado servlets. El servidor web Apache es parte del motor de Tomcat, este a su vez utiliza como contenedor de servlets Catalina.

En la actualidad, Tomcat es usado como servidor Web autónomo en entornos con alto nivel de tráfico y alta disponibilidad **[WEB19]**.

Secure Sockects Layer (SSL)

El protocolo SSL es un sistema diseñado por Netscape Communications Corporation. Se encuentra en la pila OSI entre los niveles de TCP/IP y los protocolos HTTP, FTP y LDAP entre otros. El servicio de seguridad que proporciona es el cifrado de datos intercambiados entre el cliente y el servidor, a través de un algoritmo de cifrado simétrico **[WEB20]**.

SSL proporciona autenticación y privacidad de la información entre extremos sobre Internet. Generalmente sólo el servidor es autenticado, garantizando así su identidad, mientras que el cliente se mantiene sin autenticar, denominándose una autenticación simple. En el caso de una autenticación mutua entre el cliente y el servidor, se necesitaría un despliegue de infraestructura de claves públicas PKI en la parte del cliente, como es el caso del Documento Nacional de Identificación Electrónica (DNIe). El protocolo de acción de SSL se resume en las siguientes fases:

1. **Negociar entre las partes el algoritmo que se usará en la comunicación.** Entre estos algoritmos criptográficos destacar la criptografía de clave pública (RSA y DSA, entre otros) y cifrado simétrico (RC2, RC4, IDEA, DES, AES), con funciones de hash (MD5 o derivados de SHA).

2. **Intercambio de claves públicas y autenticación basada en certificados digitales.**
   Los certificados digitales, son un documento digital mediante el cual un tercero confiable, generalmente una autoridad de certificación, garantiza la vinculación entre la identidad del sujeto y su clave pública. Los formatos de los certificados digitales son muy variados, pero los más comúnmente empleados se rigen por el estándar UIT-T X.509. Un certificado esta formado usualmente por el nombre de la entidad certificadora, el número de serie, la fecha de expiración, una copia de la clave pública del titular del certificado y la firma digital de la autoridad emisora del certificador, por ejemplo la Fábrica Nacional de Moneda y Timbre (FNMT).

3. **Cifrado del tráfico basado en cifrado simétrico.**
   El método criptográfico simétrico, se basa en que usa una misma clave para cifrar y descifrar mensajes. Ambas partes se ponen de acuerdo de antemano sobre la clase a usar.

HTTPS y SSL utilizan aproximaciones distintas con el fin de proporcionar comunicaciones seguras. SSL, como se comentó anteriormente, ejecuta un protocolo de seguridad en el momento de establecer una conexión segura a nivel de socket, a través del nombre de la máquina y el puerto, siendo estos servicios transparentes al usuario y a la aplicación. El protocolo HTTPS, es la combinación de los protocolos Hypertext Transfer Protocol (HTTP) con SSL/TLS (Transport Layer Security sucesor de SSL). La comunicación segura se negocia a través de las cabeceras y atributos de la página, por lo que los servicios de HTTPS están disponibles sólo para conexiones de HTTP **[WEB 21]**. Con el objetivo de situar las diferentes capas que intervienen, se ha optado por representarlo en el diagrama correspondiente a la ilustración 63.

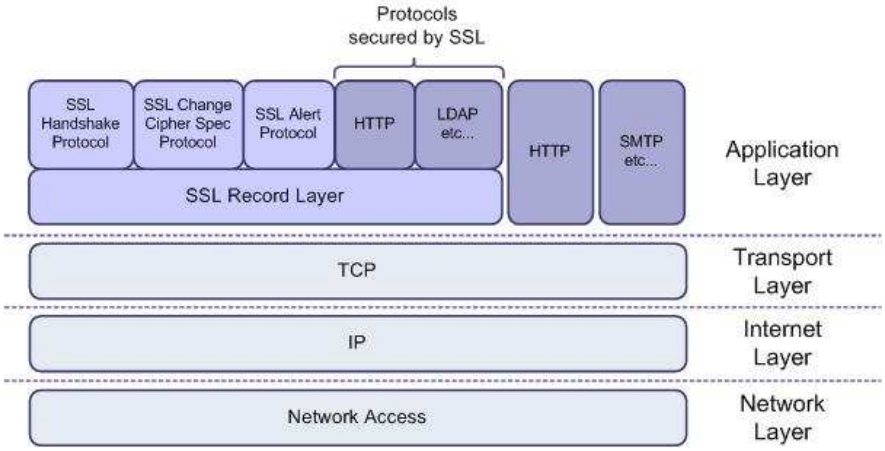


Ilustración 63 Protocolo SSL

Secure Shell (SSH) y Secure Copy (SCP)

Es un protocolo y un intérprete de órdenes seguras, se usa para acceder a máquinas remotas a través de la red. Permite ejecutar órdenes sobre la máquina conectada mediante un terminal remoto. Además de la conexión a otros dispositivos, SSH permite copiar datos de forma segura por un canal tunelizado, simulando sesiones de FTP cifradas. El mecanismo de SSH es similar al de Telnet, la principal diferencia es el uso de técnicas de cifrado que hacen que la información que viaje por el medio vaya de manera no legible, sin que una tercera persona descubra el usuario y la contraseña de la conexión establecida.

SCP es un medio de transferencia segura de ficheros informáticos entre una máquina local y una remota o entre dos máquinas remotas, usando el protocolo SSH, escenificado en la ilustración 64. Los datos son cifrados durante su transferencia **[WEB22]**.

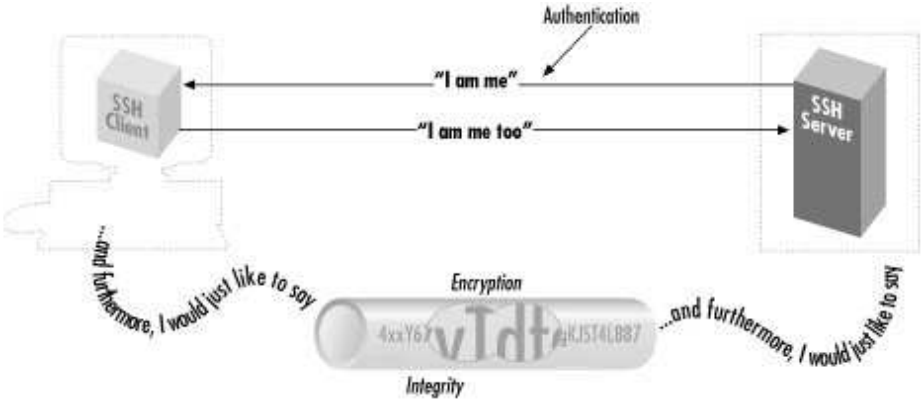


Ilustración 64 Escenificación de una comunicación SSH