Intercambio distribuido de alertas para la gestión de ataques coordinados

Joaquín García-Alfaro Ignasi Barrera-Caparròs

Departament d'Enginyeria de la Informació i les Comunicacions,
ETSE, Universitat Autònoma de Barcelona,
Edifici Q, 08193 Bellaterra, Barcelona,
Email: {jgarcia,ibarrera}@deic.uab.es

Resumen Los ataques e intrusiones contra sistemas informáticos suponen a diario grandes pérdidas de ingresos en sitios y aplicaciones de comercio electrónico. La prevención de estos ataques no es posible considerando únicamente información aislada desde los distintos elementos de una red. Una visión global del entorno al completo es necesaria para poder detectar y reparar las distintas acciones que componen dichos ataques. Actualmente estamos trabajando en una plataforma para la gestión de ataques e intrusiones informáticas, donde la cooperación entre los distintos componentes de esta plataforma ha sido solucionada de manera eficiente mediante el uso de un modelo publicador/suscriptor. En este artículo presentamos las ventajas que aporta el uso de este paradigma de comunicación para la comunicación de nuestros componentes, así como el diseño de una infraestructura específica para nuestra propuesta. Por último, apuntamos a una primera implementación y evaluación de esta infraestructura, desarrollada para sistemas GNU/Linux.

Palabras clave: Sistemas de Detección de Intrusos, Modelo Publicador/Suscriptor, Paradigmas de Comunicación, Mensajes IDMEF

1. Introducción

Cuando un atacante se hace con el control de una red informática, comprometiendo sus usuarios, equipos o aplicaciones, la red y sus recursos pueden acabar siendo utilizados como parte activa de un ataque mayor como, por ejemplo, un escaneo de puerto coordinado, o una denegación de servicio distribuida contra terceras partes. Ambos tipos de ataques, tanto coordinados como distribuidos, se basan en la realización de acciones elementales, llevadas a cabo por un adversario que trata de violar la política de seguridad de un sistema remoto. La prevención de este tipo de ataques no es sencilla, ya que requiere una recogida de información global, que permita la generación de una vista al completo del sistema a proteger. Por ello, diferentes eventos deberán ser recogidos y puestos en común, en busca de conexiones sospechosas, iniciación de procesos anómalos, modificación de información confidencial, etcétera.

Actualmente estamos trabajando en el diseño y desarrollo de una plataforma para la detección y gestión de este tipo de ataques. Nuestra propuesta se basa en la recogida de información por parte de los distintos elementos de protección de la red, junto con su correspondiente puesta en común en un proceso de correlación de alertas [3]. A través del desarrollo de una infraestructura cooperativa basada en un paso de mensajes, los distintos componentes de nuestra plataforma tratarán de colaborar en el proceso de descubrimiento y reparación de los ataques detectados. De esta forma, será posible tratar de prevenir la finalización de estos ataques, tan pronto como las primeras fases o los conjuntos de acciones elementales de estos ataques sean descubiertos a través del proceso de correlación.

En este artículo presentamos una extensión al esquema de comunicación propuesto para nuestra plataforma en [4], así como una evaluación de la infraestructura implementada para el intercambio de información de nuestros componentes. Esta infraestructura está especialmente diseñada para permitir la colaboración entre los distintos elementos de la plataforma, de manera que puedan cooperar en las distintas etapas del proceso de correlación descentralizado.

El resto de este trabajo se ha organizado de la siguiente manera. La sección 2 abre el artículo con una breve introducción al modelo de comunicación publicador/suscriptor, presentando su conveniencia para la realización de nuestro trabajo, así como algunos antecedentes relacionados con nuestra propuesta. En la sección 3, discutiremos el desarrollo de nuestra infraestructura de comunicación, basada en dos tecnologías publicador/suscriptor. La sección 4 presenta brevemente el desarrollo de un primer prototipo de nuestra propuesta, y una primera evaluación de la misma. Por último, la sección 5 cerrará nuestro artículo, con una serie de conclusiones y líneas de continuidad al trabajo presentado en este documento.

2. Modelo de comunicación publicador/suscriptor

El modelo publicador/suscriptor está especialmente pensado para situaciones de comunicación en grupo, es decir, situaciones donde un mensaje (o *notificación*) es enviado por una única entidad y es requerido por, o distribuido para, múltiples entidades. Generalmente es utilizado para la diseminación de información de manera cómoda y eficiente entre distintos elementos de un mismo grupo. Al contrario que en un modelo de comunicación *multicast*, los clientes del modelo publicador/suscriptor tienen la posibilidad de describir los eventos en los que están interesados de una forma más específica (por ejemplo, basándose en el contenido o en las cabeceras de una notificación).

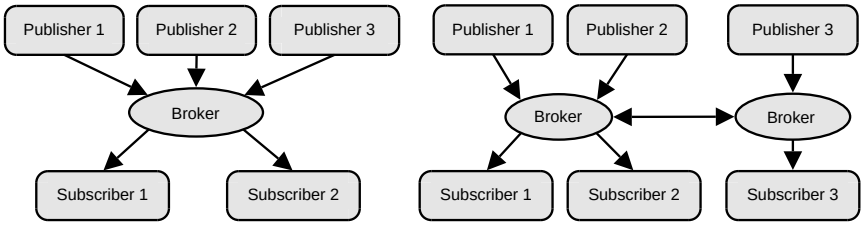
2.1. Sistemas publicador/suscriptor

Un sistema publicador/suscriptor consiste en, al menos, un encaminador que redirige las notificaciones publicadas por parte de los clientes del sistema, hacia otros clientes que

han mostrado su interés en recibir dichas notificaciones. Por razones de escalabilidad, es normal la implementación de una red distribuida de encaminadores para gestionar el *servicio de notificaciones*, es decir, el servicio que hace posible la entrega de mensajes a través de dicho sistema. De esta forma, el servicio proporcionará una infraestructura distribuida para el encaminamiento de notificaciones (incluyendo además las notificaciones necesarias para la gestión y mantenimiento del propio servicio de distribución de mensajes).

La entrega de las notificaciones puede realizarse de manera síncrona o asíncrona. Para ello, los clientes deben suscribirse de la forma adecuada al encaminador, de manera que éste pueda vincular con el cliente aquellas notificaciones entregadas por publicadores del sistema. En cuanto un encaminador reciba una nueva notificación, se encargará de comprobar y verificar sus suscripciones locales y, en caso de encontrar coincidencias, de entregarla al cliente o clientes que lo hayan solicitado. Adicionalmente, el encaminador se encargará de reenviar también las nuevas notificaciones a otros encaminadores que convivan en el mismo grupo (es decir, aquellos que estén configurados para actuar como repetidores del encaminador actual). Para más información, recomendamos la lectura de [9] para una introducción en mayor profundidad sobre el tema.

Un ejemplo de sistema publicador/suscriptor mono-encaminador es el mostrado en la figura 1(a). Aquí, cinco clientes están relacionados con un único encaminador de la manera siguiente. Tres clientes están publicando notificaciones y otros dos clientes están suscritos a un subconjunto de las notificaciones publicadas en este encaminador. Los suscriptores pueden decidir suscribirse o cancelar sus suscripciones de manera independiente y autónoma. El encaminador se encargará de hacer llegar las notificaciones recibidas a los clientes que así lo deseen, garantizando de esta manera que cada notificación es entregada a todos los suscriptores interesados.



(a) Entorno mono-encaminador (b) Entorno multi-encaminador

Figura 1. Dos ejemplos de sistemas publicador/suscriptor.

Este sencillo ejemplo puede ser extendido mediante la utilización de múltiples encaminadores configurados en forma de *cluster* (ver figura 1(b)), de manera que puedan intercambiar los mensajes recibidos, y encaminarlos hacia su destinatario final. Este nuevo diseño permite que clientes suscritos a uno de los encaminadores del sistema reciban mensajes que han sido publicados sobre otro encaminador, haciendo transparente de cara al cliente final el camino necesario para obtener estos mensajes.

La puesta en funcionamiento de este esquema es transparente para el desarrollador de aplicaciones publicador/suscriptor, ya que será el administrador del sistema quien deberá realizar la configuración del cluster. De esta manera, una aplicación podrá tener un enfoque centralizado o descentralizado, según la configuración del conjunto de encaminadores. Los suscriptores deberán únicamente realizar la petición de información a recibir, pudiendo elegir entre distintas metodologías, según el sistema publicador/suscriptor seleccionado. Dos de las metodologías más comunes son, respectivamente, la suscripción basada en temas (*topic-based subscription*) y la suscripción basada en contenidos (*content-based subscription*).

Una metodología de suscripción basada en temas es más sencilla de manejar, en comparación a una metodología de suscripción basada en contenido. Aquí, los suscriptores especifican su interés en un tema, y recibirán todos los mensajes publicados sobre este tema. Uno de los mecanismos generalmente utilizados para la combinación de mensajes suscritos y publicados es la utilización de un esquema basado en canales, de manera que una suscripción será combinada con una publicación en el sistema, si dicha suscripción coincide exactamente con la temática del canal donde dicho mensaje ha sido publicado. Otros mecanismos más completos, como la gestión de temas de forma jerárquica, pueden ser también aplicados. En este segundo caso, las suscripciones podrán ser combinadas con la temática de un canal, o con aquellos subconjuntos de mensajes que derivan de dicho canal.

Por último, las suscripciones basadas en contenido permiten la realización de suscripciones de manera más sofisticada. Por el contrario, el coste en el rendimiento del sistema es mayor que la solución anterior, en especial a la hora de realizar las tareas de encaminamiento de los mensajes. Aquí, una suscripción puede ser formulada de manera más precisa, basándose en combinaciones realizadas sobre el contenido del mensaje a recibir, mediante la utilización de lenguajes basados en expresiones regulares o consultas que pueden llegar a ser extraordinariamente complejas.

2.2. Antecedentes

Las soluciones actuales para la comunicación de componentes en plataformas de gestión de ataques son, mayoritariamente, arquitecturas cliente/servidor que implementan esquemas de comunicación centralizados o jerárquicos. Estas soluciones presentan problemas relacionados con la saturación del servicio de comunicación por parte de analizadores centrales o del nivel superior de las jerarquías. Sistemas centralizados como DIDS [12] y NADIR [6], que basan la correlación de la información en elementos dedicados, presentan fuertes restricciones en cuanto a escalabilidad y eficiencia. Éstos se encargan de almacenar toda la información recogida por el sistema en un punto central para proceder a la ejecución del proceso de detección. Esquemas jerárquicos, como los utilizados por GrIDS [13] y NetSTAT [15] continúan utilizando nodos dedicados que actúan como puntos centrales de recogida de información. Aunque estos esquemas mejoran la escalabilidad respecto a esquemas centralizados, siguen presentando defi-

ciencias similares y son vulnerables a problemas de saturación de servicio por parte de los nodos superiores de la jerarquía.

En contra de estos esquemas tradicionales, la mayor parte de propuestas alternativas tratan de eliminar la necesidad de utilizar elementos dedicados. La idea de distribuir el proceso de detección, mediante la colaboración de distintos componentes de la plataforma de detección, presenta grandes mejoras respecto a esquemas centralizados y jerárquicos. La principal ventaja de este nuevo diseño es la inexistencia de puntos de fallo aislados o cuellos de botella en el sistema.

Algunos diseños basados en paso de mensajes como, por ejemplo, CSM [16] y Quicksand [7], tratan de eliminar esta necesidad de elementos centrales o jerárquicos mediante el uso de una arquitectura P2P (*Peer-to-Peer*). Así, en lugar de disponer de estaciones centrales o jerárquicas, plantean la utilización de entidades de trabajo distribuidas por los diferentes entornos a vigilar. Tales entidades tratan de realizar tareas equivalentes de correlación mediante esquemas cooperativos. Para ello, compartirán la información de sus respectivos procesos de detección mediante la utilización de una infraestructura de comunicación común que permitirá la realización de un algoritmo de correlación descentralizado.

Desde nuestro punto de vista, estos diseños parecen apuntar en la buena dirección para la implementación de arquitecturas descentralizadas para la gestión de ataques e intrusiones. Sin embargo, las propuestas anteriores presentan un diseño muy limitado y sufren de algunas limitaciones. En la mayor parte de esquemas alternativos estudiados, cada elemento requerirá un conocimiento global del sistema, de manera que todos los elementos han de estar conectados entre sí, y mantener de forma local la matriz de conexiones necesaria para poder comunicar la información con el resto. Esto hace que el rendimiento de dicho sistema sea extremadamente costoso de controlar y mantener, presentando una escalabilidad mínima.

Otra desventaja que aparece en la mayor parte de los esquemas anteriores, reside en la necesidad de conocer, a priori, la ruta necesaria para encaminar las notificaciones a transmitir (de forma similar al servicio ofrecido por un gestor de mensajería). Por este motivo, si el número de destinos posibles crece, la gestión del servicio de comunicación de alertas será extremadamente compleja, presentando, nuevamente, una escalabilidad mínima. Otros inconvenientes aparecen en aquellos diseños que basen su servicio de intercambio de información en sistemas de inundación, los cuales facilitan el mantenimiento de gestión, pero complican el rendimiento y escalabilidad, pues la complejidad en el intercambio de mensajes crecerá de forma proporcional al número de encaminadores intermedios en el sistema.

La mayor parte de estas limitaciones puede ser solucionada de manera eficiente mediante el uso de una infraestructura basada en un modelo publicador/suscriptor. La principal ventaja de utilizar este modelo, en comparación a los paradigmas de comunicación anteriores, radica en la separación entre productores y consumidores de la información a intercambiar.

Así, podemos evitar los problemas directamente relacionados con la escalabilidad y gestión de la infraestructura de comunicaciones, por medio de la diferenciación de roles, es decir, utilización de publicadores, encaminadores y suscriptores. Un publicador, dentro de dicho esquema, no tendrá que tener ningún conocimiento sobre las entidades que acabarán por consumir la información que va a divulgar sobre el sistema.

De igual modo, los suscriptores de tal información no necesitarán tener ningún conocimiento a cerca de la ubicación física de los productores de la información. De esta forma, la inclusión de nuevos servicios en la plataforma de comunicaciones podrán ser añadidos sin necesidad de influir en el rendimiento del sistema, ni la interrupción o modificación de servicio o clientes ya existentes.

3. Nuestra propuesta

Esta sección describe los elementos de nuestra plataforma de gestión de ataques, su interacción y las tecnologías publicador/suscriptor escogidas para realizar la comunicación entre los distintos componentes. La siguiente figura muestra una vista rápida sobre nuestra propuesta, y los elementos que serán descritos a continuación.

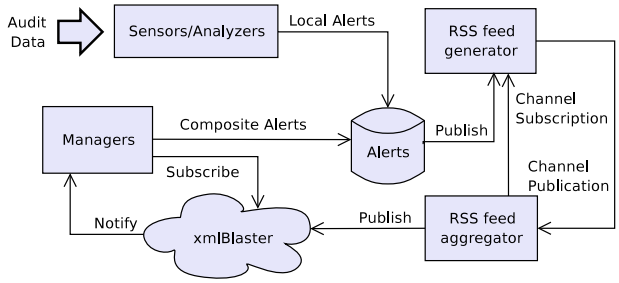


Figura 2. Esquema de nuestra propuesta.

Analizadores – Los analizadores (junto con sus respectivos sensores o unidades de detección) son los elementos encargados de notificar alertas locales a partir de un conjunto de información de auditoría local. Su tarea consiste en analizar la información recogida por sensores de red o de sistema, y almacenarla en una base de datos de alertas en formato IDMEF [2].

El formato IDMEF (*Intrusion Detection Message Exchange Format*) es una propuesta para el intercambio de mensajes entre los distintos componentes de dispositivos de seguridad, tales como sistemas de detección de intrusos (*Intrusion Detection Systems*, IDS), sistemas cortafuegos (*Firewalls*), etcétera. Cada notificación en formato IDMEF presentará una lista de atributos con sus respectivos tipos. De esta forma, a partir de

la notificación será posible identificar el analizador que creó la alerta (*AnalyzerID*), la marca de tiempo en la que fue creada (*CreateTime*), la marca de tiempo del evento recogido por el sensor (*DetectTime*), el origen y destino del evento, etcétera. Además, cada alerta proporcionará una clasificación que definirá de forma única el motivo por el cual la alerta ha sido creada. Esta clasificación será conocida por el resto de componentes del sistema, de forma que cualquiera de ellos será capaz de instanciarla en su base de conocimiento, haciendo posible un proceso correlación de información a través de estas alertas [3].

Gestores – La utilización de múltiples analizadores en un entorno distribuido, con distintas técnicas de detección en sus unidades de recogida de eventos (sensores), incrementa las probabilidades de detectar un mayor número de indicios de ataque, pero dificulta, a su vez, el proceso de gestión y tratamiento de alertas, tanto locales como externas al entorno donde estén instalados los distintos componentes. Por ello, es necesario la utilización de un conjunto de gestores de alertas encargados de realizar, de la forma adecuada, un proceso de fusión, agregación y correlación de alertas [3].

Durante el proceso de fusión, todas aquellas alertas que apunten a un mismo evento, reportado por distintos analizadores de la plataforma, serán agrupadas y transformadas, en un proceso posterior de agregación, en una única alerta que identificará de forma unitaria el evento. A continuación, la realización del proceso de correlación de alertas propuesto para nuestra plataforma (descrito en [3]) permitirá encontrar enlaces lógicos que relacionen distintas alertas de un mismo escenario de ataque, apuntando hacia el objetivo perseguido por el atacante, y posibles contramedidas que puedan detenerlo.

3.1. Publicación y suscripción de alertas locales basada en canales

El conjunto de alertas almacenadas en la base de datos de cada entorno de detección, puede ser visualizado y exportado al resto de componentes a partir de un esquema de sindicación de alertas basado en RSS (*Really Simple Syndication*) [5]. RSS es un sistema publicador/suscriptor basado en canales, muy utilizado en la actualidad por multitud de servicios y bitácoras web (*weblogs*) para la diseminación de noticias y opiniones a través de Internet. La arquitectura de RSS es realmente simple: los clientes realizan suscripciones sobre canales RSS (*RSS feeds*) sobre los que se publicarán informaciones en la que están interesados, y de manera periódica estos clientes irán realizando sondeos sobre estos canales para recibir las novedades. El contenido de RSS, directamente generado a partir de las alertas IDMEF de la base de datos local, es codificado y organizado mediante lenguaje XML. De esta forma, las alertas IDMEF podrán ser integradas de forma unitaria mediante cualquier agregador de canales RSS (ver figura 3).

A través de un agregador de canales RSS específicamente diseñado para nuestra plataforma, las alertas publicadas a través del generador de canales RSS serán de nuevo publicadas, en caso de que el administrador así lo desee, sobre el segundo sistema publicador/suscriptor que introducimos en el siguiente apartado. Las alertas publicadas sobre este segundo sistema publicador/suscriptor, más completo y eficaz a la hora de

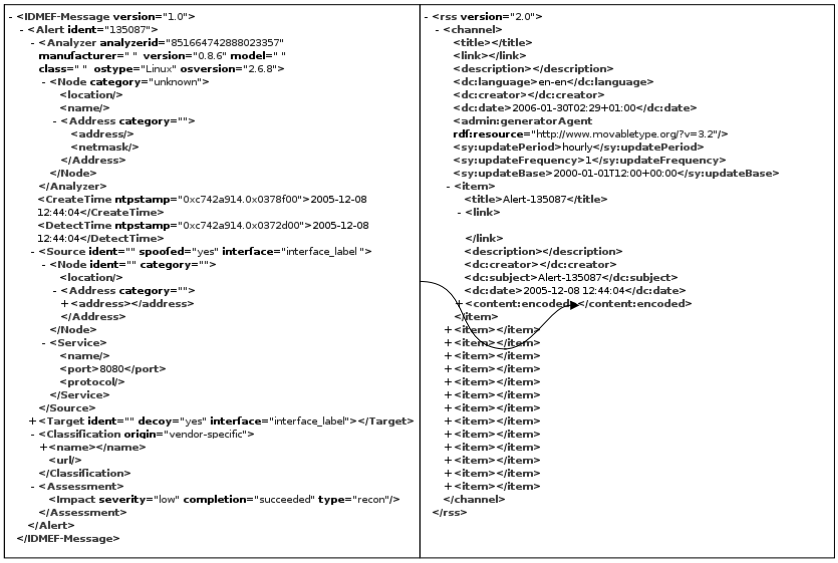


Figura 3. Integración de una alerta IDMEF local en un canal RSS.

realizar búsquedas y suscripciones, podrán ser accedidas por los gestores del resto de entornos de la plataforma, haciendo posible las tareas de cooperación necesarias para llevar a cabo el proceso de detección descentralizado presentado en [3]. Este agregador de canales RSS específico de cada entorno se encargará de realizar las suscripciones, sondeos y publicaciones necesarias, según la política de administración configurada por el responsable de la plataforma.

3.2. Publicación y suscripción de alertas globales basada en contenido

Como introducíamos en el apartado anterior, la comunicación entre los distintos gestores de la plataforma, que realizan las tareas de cooperación y correlación descentralizada de alertas, es proporcionada por *xmlBlaster*, un sistema publicador/suscriptor *open source* basado en contenido [11].

Las alertas son nuevamente expresadas en XML, ya que este es el formato utilizado para el intercambio de información en xmlBlaster. Cada mensaje consiste en una cabecera, a través de la cual se aplicarán los filtros de xmlBlaster, el cuerpo del mensaje (donde reside el mensaje en sí) y una sección de control (para tareas internas de gestión).

Los filtros son formulados mediante expresiones en *XPath*, que serán evaluadas sobre las cabeceras de los mensajes, para que cada encaminador pueda decidir sobre que suscriptores ha de ser entregado el mensaje.

Las posibles interacciones ofrecidas por parte de xmlBlaster a los clientes conectados al sistema son mostradas en la figura 4. Para publicar alertas, los clientes de la plataforma invocarán la operación de entrada *pub*(a), pasando la alerta a como parámetro. Una vez publicada, la alerta será entregada por el encaminador del sistema a los suscriptores adecuados, a través de la operación de salida *notify*(a).

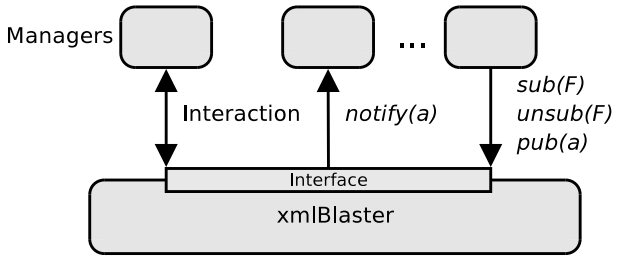


Figura 4. Interacción de los gestores a través de xmlBlaster.

Los suscriptores expresarán su interés en recibir alertas específicas, mediante la utilización de la operación *sub*(F), siendo F el filtro (o expresión XPath) pasado como parámetro a la operación. Cada cliente podrá tener múltiples suscripciones activas, las cuales podrán ser revocadas de manera independiente a partir de la operación *unsub*(F).

Todas estas operaciones son instantáneas y toman sus argumentos a partir del conjunto de clientes $\mathcal{C}$, del conjunto de alertas $\mathcal{A}$ y del conjunto de filtros $\mathcal{F}$. De manera formal, podemos expresar un filtro $F \in \mathcal{F}$ como la siguiente expresión booleana:

$$F: \quad a \longrightarrow \{\text{true}, \text{false}\} \qquad \forall a \in \mathcal{A}$$

De esta forma, una *notificación* n *aplica sobre un filtro* $F \in \mathcal{F}$ ssi $F(a) = \text{true}$. Asumimos que una alerta será publicada de forma única sobre el sistema y que un filtro será asociado a una alerta mediante un identificador único, de manera que el encaminador encargado de la entrega será capaz de identificar tal asociación.

Para nuestro trabajo, los distintos gestores de cada entorno podrán registrar su interés en un subconjunto $\mathcal{A}$ de las alertas publicadas sobre la infraestructura de alertas invocando la operación *sub*(A), que tomará el filtro A como parámetro, con

$$A(a) = \begin{cases} \text{true} & , a \in \mathcal{A} \\ \text{false} & , a \notin \mathcal{A} \end{cases}$$

Una vez suscritos a estos filtros, el sistema entregará a los gestores aquellas alertas, publicadas a través del agregador de RSS, que sean aplicables a estos filtros.

4. Evaluación de un primer prototipo

En esta sección presentamos la evaluación de CIDEX (*Communication Infrastructure for a Decentralized Exchange of Alerts*), un prototipo software que implementa la infraestructura de comunicación de alertas propuesta en este artículo. Para la evaluación de CIDEX, se realizaron las pruebas que detallamos en los siguientes apartados. Estas pruebas fueron llevadas a cabo en un equipo con procesador Pentium M a 1.4 GHz, con 512 MB de memoria RAM, funcionando sobre un sistema operativo Debian GNU/Linux 2.6, configurado con un servidor de HTTP Apache/1.3, un intérprete PHP/4.3 y un cliente Java HotSpot VM 1.5 – para la ejecución de xmlBlaster.

La generación de alertas en forma de mensajes IDMEF se realizó a partir de un conjunto de analizadores gestionados por *prelude* [14] (entre ellos, *snort* [10], compilado para ser integrado como analizador de prelude en nuestra plataforma). En este caso, Las alertas generadas por snort, fueron notificadas a prelude, y almacenadas por éste sobre una base de datos PostgreSQL. Para la publicación de las alertas de esta base de datos sobre HTTP, fue desarrollado un generador de canales RSS mediante lenguaje PHP [1]. La comunicación de componentes mediante xmlBlaster se basó en la librería de sockets en C para xmlBlaster. Por último, el análisis de mensajes IDMEF por parte de los suscriptores de xmlBlaster fue desarrollado a partir de la librería *libidmef* [8].

Consumo de CPU y espacio de memoria ocupado – Los resultados en cuanto a consumo de CPU de los encaminadores de xmlBlaster, así como de publicadores de alertas (ver figura 5), se mantuvo estable en todo momento alrededor del 2 % y 4 %, mientras el número de alertas fue inferior a 1000. A partir ese momento, se constató que el consumo de CPU por parte de los publicadores se empezaba a elevar. En cambio, el consumo de los suscriptores, recibiendo notificaciones de forma asíncrona a través de xmlBlaster, se mantuvo estable en todo momento entre el 0 % y el 1 %.

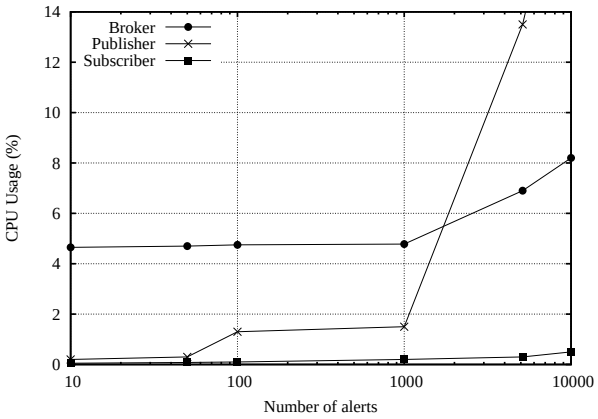


Figura 5. Utilización de CPU.

Los resultados en cuanto a consumo de memoria (ver figura 6) muestran que suscriptores y encaminadores mantuvieron un consumo estable entre el 1 % y 15 % (respectivamente) en todo momento, mientras que el consumo de memoria por parte de publicadores se empezó a elevar de forma exponencial a partir de las 100 alertas publicadas.

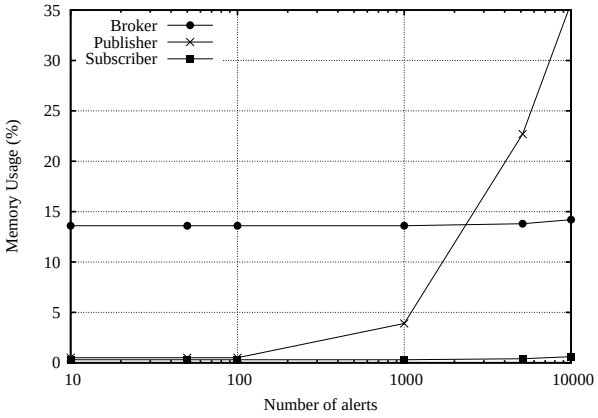


Figura 6. Espacio de memoria utilizado.

Como veremos en el siguiente apartado, este incremento desmesurado en el consumo de CPU y memoria por parte de los publicadores no se debe a las funciones específicas de xmlBlaster, sino que es directamente proporcional al consumo necesario para la descarga de notificaciones RSS, y el procesado de los mensajes IDMEF que residen en su interior. Por ello, concluimos que en el caso de utilizar CIDEX para la comunicación de alertas en un escenario de detección real, el consumo de memoria y de CPU por parte de las funciones internas a xmlBlaster para encaminamiento, suscripción y publicación, se mantendrán estables de manera independiente al flujo de alertas.

Rendimiento del agregador de canales RSS – Esta evaluación se basó en medir la latencia del agregador de canales RSS a la hora de recibir, analizar y publicar las alertas sobre xmlBlaster. Los resultados que se presentan en la figura 7 muestran como esta latencia depende proporcionalmente del número de alertas a agregar. Esto es debido a que, durante la realización de esta evaluación, las alertas fueron publicadas sobre xmlBlaster de manera proporcional a las alertas que fueron recibidas por el agregador, sin tener en cuenta los consumos relativos a las funciones de fusión y agregación de alertas IDMEF – inherentes al rol específico de este componente en la plataforma presentada en [3]. No se creyó tampoco necesario tomar los consumos de CPU y memoria del generador y agregador de RSS, ya que estos consumos de rendimiento dependerán directamente del servidor de HTTP utilizado (*Apache/1.3*) y de los lenguajes utilizados (*PHP/1.4* y *ANSI C*, respectivamente), así como de las funciones de fusión y agregación de alertas indicadas en [3]. Consideramos que estas medidas no son necesarias para evaluar CIDEX como infraestructura de comunicaciones de la plataforma final.

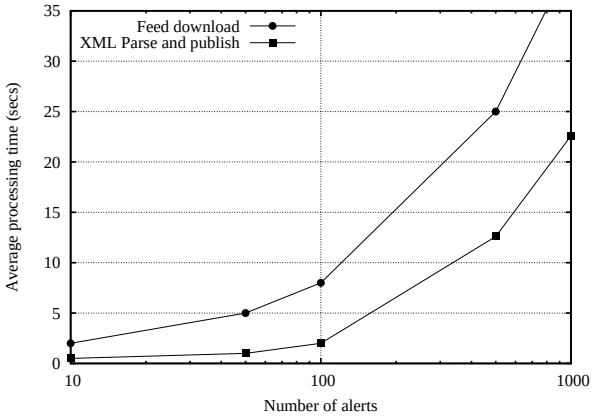


Figura 7. Evaluación del agregador de canales RSS.

Rendimiento en la entrega de notificaciones – En esta tercera prueba se analizó la latencia de notificación por parte de los encaminadores de xmlBlaster, publicando directamente y sin pasar por el agregador de RSS, a partir de tres maneras distintas de organizar las alertas publicadas: (1) árbol con un solo nodo principal (*key*) del que colgarán todos los mensajes IDMEF; (2) árbol de con tres nodos principales, realizando una organización de mensajes IDMEF a partir de tres criterios distintos; (3) árbol con un nodo principal por cada mensaje IDMEF publicado.

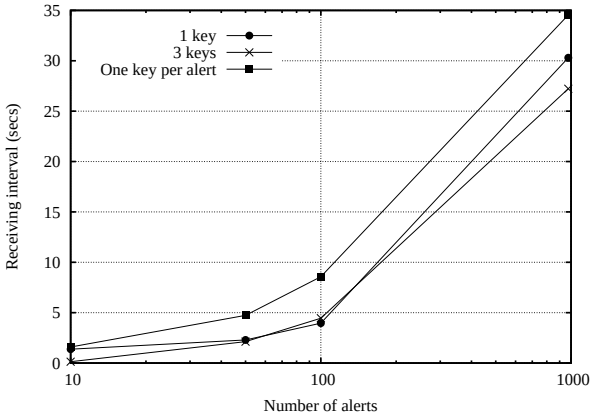


Figura 8. Evaluación de la entrega de notificaciones.

Analizando el tiempo de entrega en los dos primeros casos de organización de mensajes (ver figura 8), se pudo constatar que para un número inferior a 100 mensajes, apenas hubo diferencias. A partir de ahí, si se pudo apreciar que la latencia en la entrega de

mensajes mejora en el segundo caso de organización. Creemos que esto se debe a que la búsqueda por parte del algoritmo de encaminamiento de xmlBlaster es más sencillo en el segundo caso, ya que al existir una organización en base a tres criterios distintos, mejora la notificación final.

Por otro lado, al analizar la latencia en el tercer caso de organización de mensajes, el tiempo en la entrega a los suscriptores se resiente ligeramente. De nuevo, consideramos que esto es debido a que el algoritmo de encaminamiento de xmlBlaster se enfrenta a un número de caminos de búsqueda superior, lo cual repercute negativamente en la entrega de las notificaciones.

De esta tercera prueba se desprende también que la entrega de mensajes IDMEF mediante xmlBlaster no supuso en ningún momento un cuello de botella, pues todos los mensajes fueron procesados y entregados a tiempo, sin alcanzar en ningún momento puntos de saturación. Este resultado nos permite concluir que la utilización de xmlBlaster en un escenario de detección real garantiza las expectativas iniciales, proporcionando la escalabilidad y eficiencia esperada.

Rendimiento del sistema al completo – En esta última prueba se evaluó el rendimiento de CIDEX al completo, a partir de una generación de alertas reales por parte de los analizadores del sistema (gestionadas por prelude, y almacenadas en la base de datos del sistema PostgreSQL). Las pruebas se realizaron nuevamente para cada una de las organizaciones de mensajes comentadas durante la evaluación anterior. Como se puede observar en la figura 9, los resultados obtenidos son los esperados tras las pruebas anteriores y siguen apuntando a la segunda organización de filtros como la mejor opción. También se desprende de esta última prueba que el sistema se comporta de manera estable, siendo únicamente las funciones de descarga, análisis y generación de alertas a partir de XML/RSS los elementos que ejercen una influencia negativa en el rendimiento de la infraestructura de comunicaciones de la plataforma.

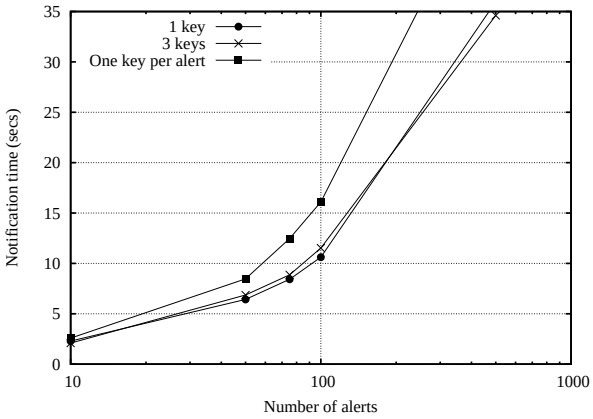


Figura 9. Evaluación de la infraestructura de comunicaciones al completo.

5. Conclusiones

En este artículo se ha presentado una infraestructura para el intercambio de información entre los distintos componentes de una plataforma para la gestión de ataques e intrusiones. Este intercambio de información, realizado a través de mensajes IDMEF [2], y basado en un modelo publicador/suscriptor, está específicamente diseñado para la realización de un proceso de correlación de alertas realizado, de forma colaborativa, entre los distintos componentes de nuestra plataforma de detección.

A diferencia de paradigmas tradicionales cliente/servidor para la comunicación de componentes, donde un enfoque centralizado o jerárquico nos lleva rápidamente a problemas de saturación y cuellos de botella, el intercambio de información entre elementos de nuestra plataforma mediante un modelo publicador/suscriptor ofrece una solución eficiente y escalable.

La utilización de las dos tecnologías publicador/suscriptor escogidas para el desarrollo de nuestra infraestructura de comunicación, canales RSS [5] y xmlBlaster [11], ha sido evaluada sobre el desarrollo de un primer prototipo para sistemas GNU/Linux. Los resultados obtenidos de esta primera evaluación muestran que el rendimiento de la infraestructura resultante cumple las expectativas iniciales y es el conveniente para realizar el desarrollo final de nuestra plataforma.

Como líneas de continuidad al trabajo presentado estamos considerando la utilización de comunicación segura entre los componentes de la plataforma a través de la utilización de SSL tanto para la comunicación local, a través de canales RSS, como para la comunicación distribuida entre entornos de detección, a través de xmlBlaster. La utilización de canales seguros a través de SSL no tan solo nos permitirá la comunicación entre los elementos de la plataforma de una forma segura, sino que hará posible la autenticación de componentes que van a interaccionar en el algoritmo de detección. Las implicaciones de la inclusión de SSL sobre nuestro esquema de comunicación, así como las repercusiones de utilizar una PKI en nuestra plataforma, deben ser aún estudiadas y evaluadas, antes de proceder a su inclusión en el presente trabajo.

Por último, se prevé iniciar un estudio sobre la introducción de técnicas de tolerancia a fallos en nuestra plataforma. Estas técnicas y mecanismos tratarán la gestión de fallos en el sistema, o partes del sistema, para reparar o reconfigurar la plataforma cuando se produzcan errores o fallos internos. Aunque algunos mecanismos de tolerancia a fallos están previstos y especificados en el sistema publicador/suscriptor utilizado para el intercambio distribuido de alertas de nuestra infraestructura (xmlBlaster), no se han podido evaluar durante la elaboración de este artículo por encontrarse aún en fase de desarrollo. Una vez estén disponibles, se procederá a su configuración y evaluación a través de la realización de nuevos experimentos sobre posibles escenarios de fallo.

Agradecimientos

El presente trabajo ha sido parcialmente financiado mediante el proyecto TIC2003-02041 del Ministerio Español de Ciencia y Tecnología, y la becas *2003FI126* y *2005BE-77* del departamento DURSI de la Generalitat de Catalunya.

Referencias

1. J. Castagnetto, H. Rawat, S. Schumann, C. Scollo, and D. Veliath. *Professional PHP Programming*. Wrox Press Inc, ISBN 1-86100-296-3, 909 pages, 1999.
2. H. Debar, D. Curry, and B. Feinstein. Intrusion detection message exchange format data model and extensible markup language. Technical report, January 2005.
3. J. García, F. Autrel, J. Borrell, S. Castillo, F. Cuppens, and G. Navarro. Decentralized publish/subscribe system to prevent coordinated attacks via alert correlation. In *Sixth International Conference on Information and Communications Security*, volume 3269 of *LNCS*, pages 223–235, Málaga, Spain, October 2004. Springer-Verlag.
4. J. García, M. A. Jaeger, G. Mühl, and J. Borrell. Decoupling Components of an Attack Prevention System using Publish/Subscribe. In *2005 IFIP International Conference on Intelligence in Communication Systems*, pages 87–98, Montréal, Canada, Springer-Verlag.
5. B. Hammersley. *Content Syndication with RSS*. O'Reilly Ed., First Edition, March 2003, ISBN 0-596-00383-8, 202 pages.
6. J. Hochberg, K. Jackson, C. Stallins, J. F. McClary, D. DuBois, and J. Ford. NADIR: An automated system for detecting network intrusion and misuse. In *Computer and Security*, volume 12(3), pages 235–248. May 1993.
7. C. Kruegel and T. Toth. Distributed pattern detection for intrusion detection. In *Network and Distributed System Security Symposium Conference Proceedings: 2002*, 1775 Wiehle Ave., Suite 102, Reston, Virginia 20190, U.S.A., 2002. Internet Society.
8. A. C. Migus. IDMEF XML library version 0.7.3. `http://sourceforge.net/projects/libidmef/`, March 2004.
9. G. Mühl. *Large-Scale Content-Based Publish-Subscribe Systems*. PhD thesis, Technical University of Darmstadt, 2002.
10. M. Roesch. Snort: lightweight intrusion detection for networks. In *13th USENIX Systems Administration Conference*, Seattle, WA, 1999.
11. M. Ruff. XmlBlaster: open source message oriented middleware. White paper [on-line]. `http://xmlblaster.org/`, 2000.
12. S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, D. M. Teal, and D. Mansur. DIDS (distributed intrusion detection system) - motivation, architecture and an early prototype. In *Proceedings 14th National Security Conference*, pages 167–176, October, 1991.
13. S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS – a graph-based intrusion detection system for large networks. In *19th National Information Systems Security Conference*, 1996.
14. Y. Vandoorselaere and L. Oudot. Prelude-IDS, un Système de Détection d'Intrusion hybride opensource. MISC issue 3, July 2002.
15. G. Vigna and R. A. Kemmerer. NetSTAT: A network-based intrusion detection system. *Journal of Computer Security*, 7(1):37–71, 1999.
16. G. B. White, E. A. Fisch, and U. W. Pooch. Cooperating security managers: A peer-based intrusion detection system. *IEEE Network*, 7:20–23, February 1999.