

# Diseño y desarrollo del subsistema de detección y reacción de una plataforma para la prevención de ataques coordinados

Sergio Castillo, Joaquín García, Joan Borrell

Dept. d'Informàtica, Universitat Autònoma de Barcelona  
Edifici Q, 08193 Bellaterra, Spain.  
{SCastillo, JGarcia, JBorrell}@ccd.uab.es

**Resumen.** Uno de los requisitos imprescindibles para el éxito en el comercio electrónico es el de proporcionar confianza y seguridad. A pesar del gran avance en este ámbito, tales como sistemas cortafuegos, mecanismos de autenticación y sistemas para la detección de intrusos, la mayoría de las infraestructuras que respaldan el comercio electrónico son actualmente vulnerables a sufrir un gran número de ataques basados en técnicas coordinadas como, por ejemplo, ataques de denegación de servicio distribuidos [5]. En este artículo presentamos una plataforma que proporciona una solución descentralizada para prevenir este tipo de ataques, y que puede ser integrada como un refuerzo complementario a los mecanismos convencionales de seguridad de cualquier institución de comercio electrónico.

## 1 Introducción

El crecimiento de Internet en los últimos años ha tenido importantes repercusiones en distintos ámbitos: economía, cultura, tecnología, etc. Del mismo modo, esta expansión ha tenido relevantes influencias en el campo de la seguridad computacional. Aún así, los sistemas conectados a Internet nunca han sido tan vulnerables. Día a día podemos hacernos eco de noticias que ponen de manifiesto el asedio al que están sometidos innumerables sistemas y las graves consecuencias que supone para el comercio electrónico.

Desde que en 1980 James P. Anderson estudiara el problema del análisis de los registros de sistema como metodología para la búsqueda de intrusos [1], las tecnologías asociadas a los sistemas de detección de intrusos han evolucionado considerablemente. A pesar de esto, el progreso no ha sido exclusivo de los procedimientos de detección, sino que también ha estado presente en las técnicas empleadas por los atacantes. Así, la utilización de métodos distribuidos y coordinados son cada día más habituales entre la comunidad de atacantes, lo que les proporciona la posibilidad de llevar a cabo acciones más complejas y difíciles de detectar.

La prevención de este tipo de ataques no siempre es posible mediante la monitorización y recogida de información aislada desde los distintos equipos de la red. En la mayoría de los casos será necesaria una visión global de los eventos y

acciones que se producen en el sistema. Por ello, la prevención y detección ante estos ataques requerirá una recogida de información específica que se encuentra repartida por los distintos equipos de la red a vigilar, y la correlación de tal información. Dentro de esta recogida de información deberemos poder obtener todos aquellos eventos y acciones que preceden a la realización del ataque, tales como establecimiento de conexiones sospechosas, ejecución de procesos anómalos, adición de información en el sistema de ficheros, cambios repentinos en el tráfico de la red, etc.

En este artículo presentamos el diseño y desarrollo del subsistema de detección y reacción de una plataforma que proporciona una solución descentralizada para prevenir ataques coordinados contra terceras partes [3]. El sistema se basa en un conjunto de entidades cooperativas (llamadas *celdas de prevención*) que son albergadas en cada uno de los recursos de red que queremos desarmar. Estas entidades colaborarán entre sí para detectar si el equipo donde se encuentran instaladas participa de forma activa en la realización de un ataque coordinado. En caso de detectar tal situación, las celdas de prevención actuarán de la forma apropiada sobre cada uno de sus recursos asociados para, finalmente, eludir su participación en el ataque detectado. Así, la principal diferencia entre nuestra propuesta frente a sistemas de detección similares es que cada nodo que alberga una celda de prevención está a la espera de ser el origen o formar parte de un ataque coordinado, no la víctima del mismo.

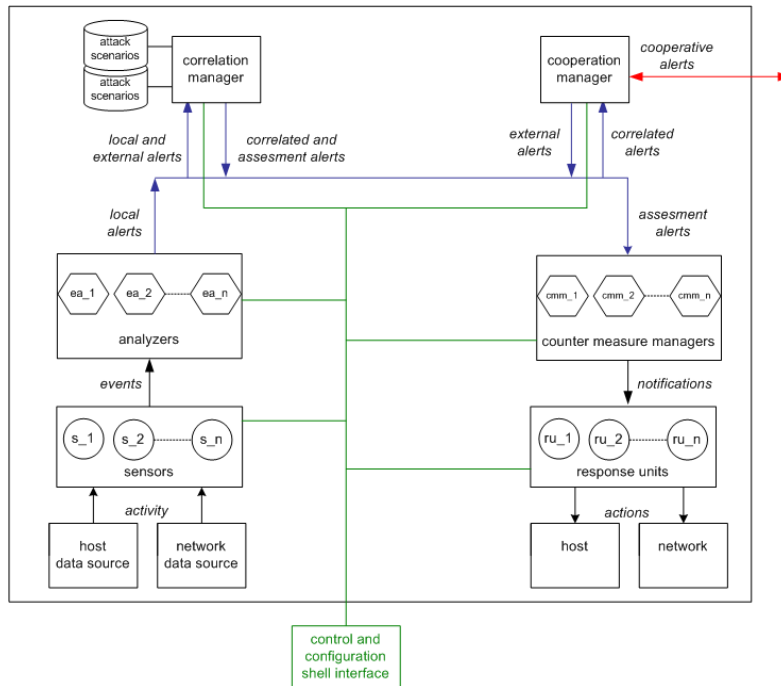
El resto del artículo está organizado de la siguiente manera. En la sección 2 presentamos una visión global de las celdas de prevención, su funcionamiento, sus elementos así como de que manera se interrelacionan. A continuación, se detalla el diseño y el desarrollo de los componentes del subsistema de detección y reacción, qué tecnologías han sido empleadas y cómo se integran dentro del modelo de celda de prevención expuesto en la sección anterior. Por último, se exponen las conclusiones que podemos desprender de nuestra propuesta así como el posible trabajo futuro de continuación.

## **2 Esquema de una celda de prevención**

En esta sección presentamos una plataforma cuyo principal propósito es la detección y prevención de ataques coordinados. El diseño de nuestro sistema tiene dos objetivos principales. El primero de ellos es obtener una arquitectura modular compuesta por un conjunto de entidades que cooperen y que denominaremos *celdas de prevención*. Estas entidades colaborarán entre sí para detectar si el equipo donde están instaladas participa de forma activa en la elaboración de un ataque coordinado contra la red donde están conectados o contra una red de terceras partes. Una vez el ataque, o una de sus acciones, haya sido detectado, estas entidades deberán ser capaces de prevenir la utilización de los recursos asociados para evitar su participación en el ataque. Para esto, será necesario que cada celda de prevención detecte actividades ilícitas locales que comunicará al resto de celdas, y que deberá correlacionar con las alertas externas en el proceso global de detección y cancelación del ataque.

El segundo objetivo es conseguir una relación completa e independiente entre los diferentes componentes que conforman cada una de estas entidades cooperativas. De esta forma, será posible distribuir tales componentes de acuerdo con las necesidades de cada recurso que queramos desarmar.

Para alcanzar la independencia entre componentes proponemos el uso de un modelo de comunicación *publicador-suscriptor*. Este modelo es de gran utilidad, no solo para la comunicación entre las distintas celdas de prevención, sino también para los componentes internos de cada una de éstas. Así pues, consideramos la utilización de un modelo *publicador-suscriptor* para la relación entre los componentes internos de cada celda de prevención. Todos estos componentes han sido diseñados de acuerdo con los elementos básicos de cualquier IDS (sensores, analizadores, gestores y unidades de respuesta). Los mensajes intercambiados entre estos componentes serán básicamente tres: *eventos* (entre sensores y analizadores), *alertas* (entre analizadores y gestores), y *acciones* (entre gestores y unidades de respuesta). Estos componentes, y los distintos mensajes intercambiados entre ellos, se muestran en la figura 1 y se organizan en dos grandes subsistemas: el de *detección y reacción*, y el de *correlación y cooperación*.



**Figura 1.** Esquema de una celda de prevención

El subsistema de detección y reacción es el responsable de realizar el proceso de detección y correlación a bajo nivel, así como el encargado de llevar a término las acciones oportunas para evitar un ataque. Podemos distinguir los siguientes elementos:

- *Sensores*, encargados de obtener información sospechosa, tanto a nivel de red como a nivel del propio equipo donde se encuentran instalados, para publicarla en el canal específico (donde algunos analizadores asociados estarán a la escucha). Proponemos la utilización de sensores basados en red (en busca de tráfico ilegal o peligroso como, por ejemplo, desbordamientos de buffer, datagramas con suplantación de IP, inundación de tráfico, etc) y de sensores basados en equipo (en busca de acciones sospechosas o peligrosas originadas en el propio equipo y que pueden suponer un riesgo como, por ejemplo, utilización abusiva de la CPU, ejecución de comandos potencialmente hostiles, etc.).
- *Analizadores*, responsables de recibir los eventos publicados por los sensores del sistema para realizar con ellos un proceso de análisis y de correlación de bajo nivel. Así, estos componentes consumirán eventos y producirán alertas locales en el interior de cada celda de prevención. Estas alertas serán publicadas en el canal correspondiente (en este caso, en el canal de alertas locales). Proponemos el uso tanto de analizadores con detección basada en usos indebidos como de analizadores con detección basada en anomalías.
- *Gestores de contramedidas*, encargados de recibir alertas de contramedida publicadas por el subsistema de correlación y cooperación. Estos gestores son los responsables de consumir contramedidas y transformarlas en las acciones correspondientes. Una vez la alerta de contramedida es transformada, las acciones asociadas serán publicadas en el canal asociado a las unidades de respuesta de la celda de prevención.
- *Unidades de respuesta*, que tomarán las acciones producidas por los gestores de contramedidas para llevarlas a cabo. Cada acción habrá sido generada para prevenir uno de los distintos pasos del ataque coordinado que ha sido detectado, y será ejecutada contra el recurso donde la celda de prevención se encuentra alojada. Al igual que los sensores, proponemos el uso de unidades de respuesta basadas en red (para, por ejemplo, bloquear conexiones, cerrar puertos TCP/UDP, etc.) y unidades de respuesta basadas en equipo (para, por ejemplo, finalizar la ejecución de procesos, realizar un cambio de permisos, bloquear cuentas de usuario, etc).

El subsistema de correlación y cooperación es el responsable de la consolidación de las alertas a través de los algoritmos de correlación, y en el cual cooperan otras celdas de prevención. Podemos distinguir los siguientes componentes:

- *Gestor de cooperación*, que estará a la escucha de alertas cooperativas provenientes del exterior de la celda de prevención donde se encuentra instalado, y publicará alertas externas en su interior. A la vez, consumirá las

alertas de correlación producidas en el interior de la celda de prevención donde se encuentra, y las publicará en el exterior en forma de alertas cooperativas. Proponemos el uso de un formato genérico para el intercambio de información entre sistemas de detección de intrusos y sistemas de respuesta, lo que permite la inclusión de componentes de terceras partes basados en la misma especificación.

- *Gestor de correlación*, que estará a la escucha de alertas locales y externas a través de los canales correspondientes, para consumir dicha información y contrastarla contra sus escenarios de ataque asociados. Este componente realizará un proceso de correlación de alto nivel. Estará, por tanto, involucrado en la parte del proceso de correlación relativa a la celda de prevención donde se encuentra albergado. Puesto que la correlación de un ataque coordinado es un conjunto de alertas internas y externas reportadas por elementos internos y externos del sistema, todas estas alertas serán analizadas en un proceso de correlación de alto nivel ejecutado por el gestor de correlación. Este elemento será también el responsable de publicar alertas de correlación y alertas de contramedida.

### **3 Diseño y desarrollo del subsistema de detección y reacción**

Esta sección muestra el diseño y desarrollo del subsistema de detección y reacción del modelo de celda de prevención expuesto en la sección anterior y que comprenden los *Sensores*, *Analizadores*, *Gestor de contramedida* y *Unidades de respuesta*. Dicho subsistema ha sido implementado para sistemas GNU/Linux en lenguaje C, concretamente se ha probado con distintas versiones de la serie 2.4.x de Linux y con las versiones 2.9.x y 3.x del compilador gcc de GNU. La naturaleza de este sistema operativo en cuanto a lo que a red se refiere, código abierto y documentación, ha facilitado el análisis de requerimientos y el desarrollo.

La principal diferencia entre nuestra propuesta frente a sistemas de detección similares reside en que cada nodo que alberga una celda de prevención está a la espera de ser el origen de un ataque coordinado (o alguna de las acciones relacionadas con dicho ataque), no la víctima del mismo. Este hecho implica algunas consideraciones en el análisis de requerimientos de los sensores y las unidades de respuesta. En primer lugar, el número de estos elementos debe ser lo suficientemente representativo para detectar y reaccionar contra los distintos pasos de los escenarios de ataque que el sistema conoce. En segundo lugar, tanto analizadores como gestores de contramedidas necesitarán una comunicación suficientemente rápida con sensores y unidades de respuesta. Es decir, esta velocidad ha de ser suficiente para poder recoger todos los eventos y proporcionar todas las respuestas posibles.

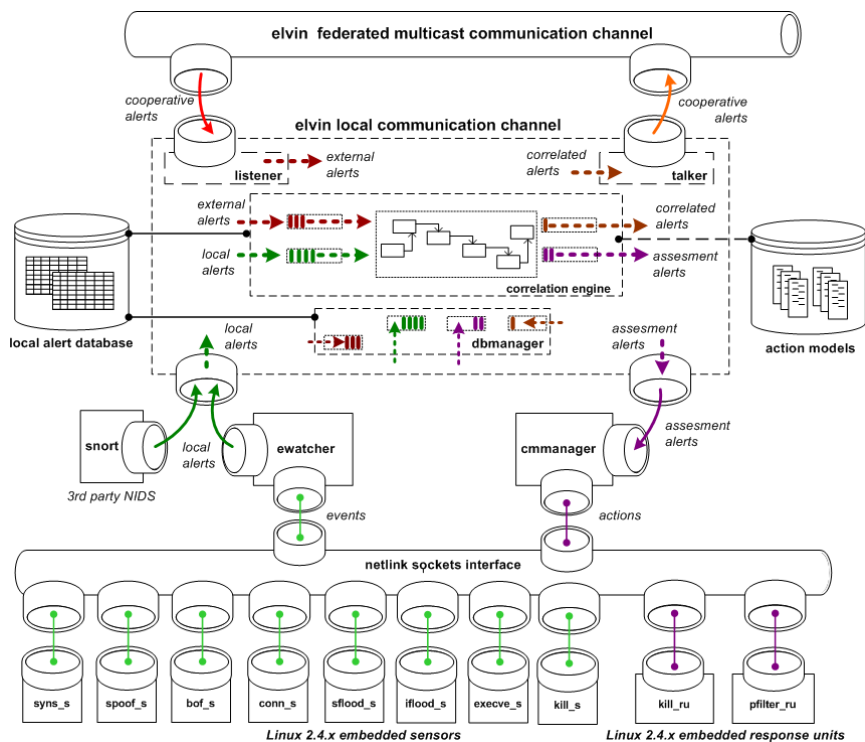
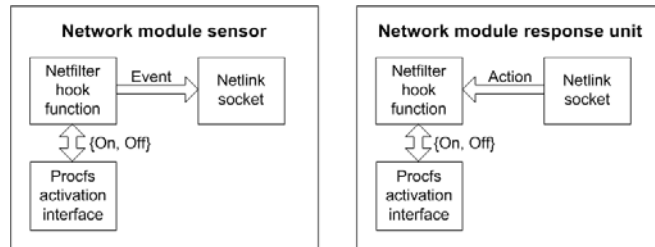


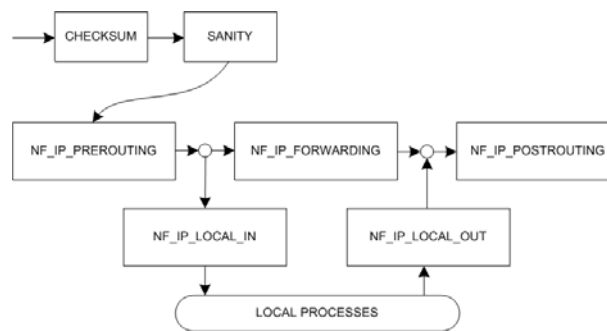
Figura 2. Esquema del desarrollo actual

### 3.1 Sensores y unidades de respuesta

Con el objetivo de conseguir los requerimientos anteriores, el desarrollo de nuestra plataforma se inició con el diseño e implementación de un conjunto de sensores y unidades de respuesta empujados en el propio kernel del sistema operativo GNU/Linux en forma de módulos. Aún así, el uso de sensores y unidades de respuesta de terceras partes es posible en nuestro sistema (tal y como muestra la figura 2 con el uso del NIDS *Snort*). La utilización de sensores y unidades de respuesta en forma de módulos de kernel ofrece a nuestro sistema toda una serie de ventajas. En primer lugar, la posición de privilegio de los módulos del kernel dentro del sistema operativo proporciona un acceso directo a cualquier información necesaria de una forma eficiente y de total confianza. Por otro lado, la carga producida por el intercambio de información entre el espacio de kernel y el espacio de usuario se reduce, transfiriendo únicamente aquella información necesaria en el momento en que un evento se ha producido. Como resultado de estas ventajas, la capacidad de proceso a la hora de analizar patrones o comportamientos (por ejemplo, datagramas IP o comandos ejecutados) puede maximizarse con facilidad.



(a) Sensores y unidades de respuesta adheridos a netfilter



(b) Esquema de la arquitectura de netfilter

**Figura 3.** Sensores y unidades de respuesta basados en red

El comportamiento y el estado tanto de sensores como de unidades de respuesta puede ser modificado dinámicamente dependiendo de los parámetros que gobiernan sus algoritmos. A través de alertas, por ejemplo, el gestor de contramedidas puede modificar el comportamiento de alguna de las unidades de respuesta añadiendo nuevas reglas de filtrado o añadiendo nuevos procesos a eliminar. El objetivo de este comportamiento es conseguir que la propia plataforma pueda adaptarse por sí misma en función del estado interno de cada celda de prevención o de la clase de ataque coordinado que se pretenda prevenir. Adicionalmente, para facilitar el acceso a estos elementos (para tareas de chequeo o de *debugging*, por ejemplo), tanto sensores como unidades de respuesta crean una entrada en el sistema de ficheros *procfs*. Este sistema de ficheros (*proc file system, procfs*) es un sistema de ficheros especial en el kernel Linux que permite acceder desde el espacio de usuario a estructuras de información dentro del espacio de kernel. En nuestro caso, tanto sensores como unidades de respuesta ofrecen la posibilidad de ser activados o desactivados a través de este sistema de ficheros virtual.

## Sensores y unidades de respuesta de red

La implementación de los sensores y de las unidades de respuesta de red (ver figura 3(a)) se basa en la utilización del subsistema netfilter [8], un marco de trabajo para la manipulación de paquetes en el kernel Linux. Netfilter permite tanto el filtrado de paquetes como la traducción de direcciones de red y otras manipulaciones más complejas. Como podemos observar en la figura 3(b), la arquitectura de netfilter contempla cinco puntos de anclaje a lo largo del protocolo de pila de IPv4, donde módulos del kernel pueden asignar funciones de retrollamada para la monitorización y filtrado de paquetes. Todo datagrama de llegada, habiendo pasado el correspondiente proceso de validación (*checksum* correcto, paquete no fragmentado, etc) pasa por el punto *NF\_IP\_PRE\_ROUTING*. Tras superarlo, el datagrama es sometido al proceso de encaminamiento, determinando si debe ser destinado a otra interfaz de red, debe ser pasado a un proceso local, o bien debe ser descartado al no existir ruta posible para éste. Si el destino del paquete es local, entra en *NF\_IP\_LOCAL\_IN* antes de ser entregado al proceso destinatario (si existe). En cambio, si el destino es un interfaz diferente pasa por *NF\_IP\_FORWARD*. El punto *NF\_IP\_LOCAL\_OUT* recibe todo los paquetes creados por los procesos locales. Por último, tanto los datagramas que han sufrido un cambio de ruta como los generados de forma local atraviesan el punto *NF\_IP\_POST\_ROUTING*. Cuando un datagrama llega a uno de los puntos descritos anteriormente, éste es pasado de forma ordenada a las funciones que hayan sido registradas por los módulos sobre aquel punto. En ese momento, la función que recibe el paquete es libre de manipularlo de diversas maneras: rechazándolo, pasándolo a la siguiente función en la cadena de funciones registradas, etc.

Puesto que el sistema de celdas de prevención trata de determinar y eliminar el origen de los ataques, todos los sensores y unidades de respuesta de red registrarán sus funciones de detección y actuación sobre el punto *NF\_IP\_LOCAL\_OUT*. Mediante un análisis de los datagramas generados localmente será posible detectar cuando una máquina forma parte de un ataque coordinado y, a su vez, actuar sobre dichos paquetes con el objetivo de eludirlo.

Actualmente se han desarrollado los sensores y unidades de respuesta de red siguientes: un sensor para la detección de exploraciones silenciosas de puerto (*syns\_s*), un sensor para la detección de IP *spoofing* (*spoof\_s*), un sensor para la detección de *buffer overflows* en paquetes de red (*bof\_s*) basado en la propuesta [7], un sensor para la detección de conexiones TCP que puede ser utilizado para inferir cadenas de conexiones TCP (*conn\_s*), dos sensores para detectar ataques de denegación de servicio (DoS) basadas en SYN e ICMP *flooding* (*sflood\_s* y *iflood\_s*) y, finalmente, una unidad de respuesta capaz de realizar filtrado de paquetes (*pfilter\_ru*).



## Sensores y unidades de respuesta de equipo

La implementación de los sensores de equipo propuestos se basa en la captura de algunas llamadas al sistema con el propósito de obtener información de interés en la búsqueda de actividades ilícitas o sospechosas. Por otro lado, la implementación de las unidades de respuesta utiliza la misma idea, proporcionando los mecanismos necesarios para prevenir las acciones necesarias asociadas con los distintos pasos del ataque a prevenir.

Para comprender la metodología empleada describiremos el funcionamiento del sensor *execve\_s*, dado que el resto de sensores y unidades de respuesta se basan en el mismo principio. Este sensor es el responsable de la monitorización de la ejecución de programas y comandos del sistema. Cuando un proceso llama a la función *execve* en espacio de usuario para ejecutar un nuevo programa, se realiza una llamada al kernel a través de la librería estándar *glibc*. Esta llamada está representada en espacio de kernel como una tabla que recibe el nombre de *syscall table*. Dicha tabla contiene una entrada por cada una de las llamadas del sistema, siendo cada entrada un puntero a la función que la implementa. De esta manera, se pone en correspondencia una llamada y su función de tratamiento asociada. En el caso que nos ocupa, el sensor modifica la entrada correspondiente a la llamada *sys\_execve*, redireccionándola a una implementación propia (figura 4), lo que le permite un control sobre ésta. El sensor intercepta de esta manera el nombre del ejecutable junto a sus parámetros, el identificador del usuario que lanzó la llamada y el identificador de proceso que se le asignará. Estos datos son enviados al analizador *ewatcher* y, posteriormente, el sensor devuelve el control a la llamada original del kernel respetando los parámetros originales de la llamada, lo que garantiza el correcto funcionamiento del sistema.

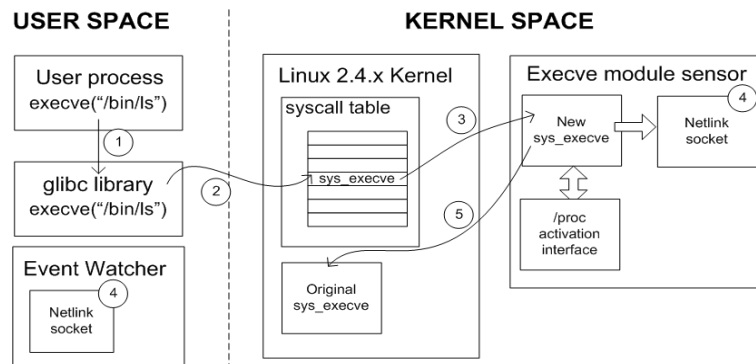


Figura 4. Captura de la llamada del sistema *sys\_execve*

Actualmente se ha implementado un sensor para monitorizar la ejecución de comandos ilícitos o sospechosos (*execve\_s*), un sensor para detectar qué procesos van a ser eliminados (*kill\_s*) y una unidad de respuesta capaz de eliminar y proteger procesos del sistema (*kill\_ru*).

### 3.2 Comunicación de eventos y acciones

Como se muestra en la figura 2, los sensores proporcionan eventos al analizador *event watcher* (*ewatcher*) y el gestor de contramedidas (*cmmanager*) provee las acciones a las unidades de respuesta. La complejidad de estos componentes y la limitación que supone trabajar en el espacio de kernel requiere su diseño como procesos demonio (*daemon*) en espacio de usuario. Así, para hacer posible este esquema, es necesario la utilización de mecanismo de comunicación entre espacio de kernel y espacio de usuario.

Entre las diferentes alternativas posibles para comunicar procesos entre el espacio de kernel y el espacio de usuario, hemos optado por la utilización de *netlink sockets* [2] para la comunicación entre los sensores y analizadores, y entre unidades de respuesta y gestores de contramedidas. *Netlink sockets* es un mecanismo específico para el kernel Linux que proporciona un enlace de comunicación bidireccional, no orientado a conexión, asíncrono y con altos ratios de transferencia. Aunque el uso de *netlink sockets* está más enfocado a la implementación de servicios IP [6], este mecanismo puede ser también utilizado como interfaz estándar para establecer un enlace de comunicación entre módulos de kernel y procesos en el espacio de usuario. Adicionalmente, *netlink sockets* se basa en las primitivas para el tratamiento de sockets tradicionales, proporcionándonos transparencia en lo que a mecanismos de *buffering* se refiere.

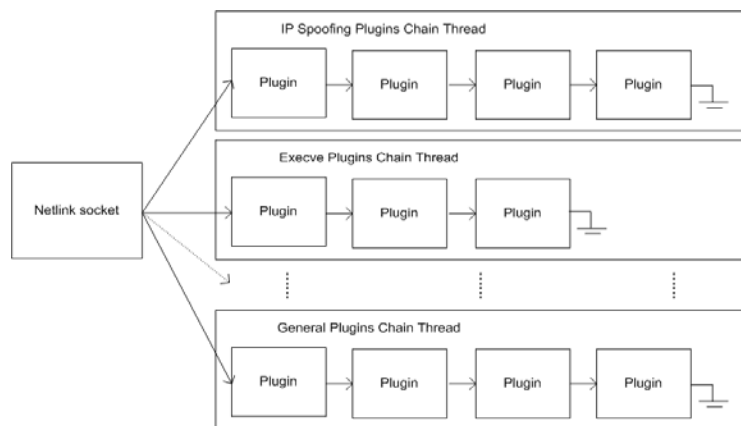


Figura 5. Arquitectura *multi-thread* del analizador *ewatcher*

### 3.3 Analizadores y gestores

Tanto la implementación del analizador *ewatcher* y del gestor de contramedidas *cmmanager*, se basan en un mecanismo de *plug-ins*, lo que facilita su desarrollo y mantenimiento. A través de *netlink sockets*, tanto analizadores como gestores de contramedidas consumen o producen información pero, para generar la información, o para manejarla, distintos *plug-ins* pueden ser encadenados. Así, gracias al mecanismo de *plug-ins*, el analizador *ewatcher* puede filtrar eventos según diversos criterios o, en el caso del gestor de contramedidas, aplicar una respuesta activa (enviando las acciones oportunas a las unidades de respuesta) o pasiva (informando a un administrador mediante una consola sin llevar a cabo ninguna acción).

Algunas de estas cadenas de *plug-ins* son lanzadas como threads. El analizador *ewatcher*, por ejemplo, lanza distintos *plug-ins* para manipular los eventos recibidos haciendo uso de un mecanismo *multi-threading* (ver figura 5). De esta forma, es posible paralelizar la recogida de los diferentes eventos producidos por el conjunto de sensores del sistema. Otros *plug-ins* como, por ejemplo, el responsable de enviar las acciones a las unidades de respuesta, o el responsable de manejar alertas externas y transformarlas a alertas locales, no necesitan el uso de este mecanismo *multi-threading* para realizar su trabajo.

## 4 Conclusiones

En este artículo hemos presentado el diseño y desarrollo de un subsistema de detección y reacción de una plataforma para la detección y prevención de ataques coordinados. Dicho subsistema, integrado dentro de entidades cooperativas que denominamos celdas de prevención, permite evitar que los recursos de las máquinas que las albergan sean usados para lanzar ataques contra terceras partes.

Gracias a la relación completa e independiente alcanzada a través de un paradigma de comunicación publicador-subscriptor, y a la concepción modular de los componentes, se ha conseguido un subsistema flexible y fácil de mantener. A su vez, el mecanismo de *plug-ins* garantiza la posibilidad de modelar respuestas activas o pasivas en función del tipo de ataque detectado.

A pesar de esto, si un atacante descubre la presencia de una celda de prevención en el sistema que utilizará, y si dispone de suficientes privilegios, intentará desactivarla previamente a la ejecución del ataque con el objetivo de evadir la detección y cancelación del mismo. De igual modo, el usuario ilegítimo podría intentar interactuar con los elementos que conforman el subsistema de detección y reacción con la finalidad de provocar un mal funcionamiento. Así, una manipulación adecuada permitiría al intruso generar alertas falsas que, enviadas a otras celdas de prevención, podrían causar denegaciones de servicio (a través de las unidades de respuesta) o bien ocultar el ataque real.

Por ello, estamos trabajando actualmente en el estudio de mecanismos que nos permitan proteger los diversos componentes de cada celda de prevención contra ataques dirigidos hacia éstas mismas. Tales mecanismos ayudarán al sistema a mitigar

o eliminar cualquier tipo de acción que atente contra los elementos de la plataforma y su correcto funcionamiento.

Respecto al mecanismo de respuesta, nuestra propuesta ofrece un soporte al administrador de los recursos o al gestor de contramedidas, quienes decidirán si la respuesta ha de ser lanzada o no. Se trata de una estrategia prudente, pero que puede llegar a introducir ciertos retrasos, incompatibles con una respuesta automática necesaria en un sistema de prevención a tiempo real. Por ello, planeamos el estudio y evaluación de nuevas técnicas de respuesta que permitan la reconfiguración de la política de seguridad del sistema afectado, tal y como se sugiere en [4]. Del mismo modo, estamos considerando estrategias seguras para la gestión global de la plataforma, sin recurrir a metodologías basadas en una administración particular de cada celda.

## Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia y Tecnología, a través de su proyecto TIC-2003-02041, y el Departamento DURSI de la Generalitat de Catalunya, a través de su beca 2003FI-126.

## Referencias

1. J. P. Anderson. *Computer security threat monitoring and surveillance*. February 1980.
2. G. Dhandapani and A. Sundaresan. *Netlink sockets overview* Technical report. The University of Kansas, September 1999.
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, October 2004.
4. M. Petkac and L. Badger. *Security agility in response to intrusion detection*. In 16<sup>th</sup> Annual Computer Security Applications Conference, New-Orleans, LA, 2000.
5. D. Moore, G. Voelker, and S. Savage. *Inferring Internet denial of service activity*. In Usenix Security Symposium, Washington, D.C, 2001.
6. J. Salim, H. Khosravi, A. Kleen, and A. Kuznetsov. *Linux Netlink as an IP Services Protocol*. Network Working Group, July 2003.
7. T. Toth and C. Kruegel. *Accurate Buffer Overflow Detection Via Abstract Payload Execution*. In Proceedings of the 5th Recent Advances in Intrusion Detection (RAID2002), Zurich, Switzerland, October 2002.
8. H. Welte, J. Kadlecik, M. Josefsson, P. M. Hardy, et. al. *The netfilter project: firewalling, nat and packet mangling for linux 2.4*, 2004.