Arquitectura Distribuida para la Respuesta Automática frente a Intrusiones en un IRS Basado en Ontologías

Guamán D.*; Mateos V.**
**Escuela Politécnica Nacional, Facultad de Ingeniería Eléctrica y Electrónica, Quito, Ecuador*
e-mail: danny.guaman@epn.edu.ec
*** Universidad Politécnica de Madrid. Escuela Técnica Superior de Ingenieros de Telecomunicación, Madrid, España*
e-mail: vmateos@dit.upm.es

Resumen: *En el presente trabajo se hace la propuesta de una arquitectura distribuida, segura y escalable para la ejecución automática de respuestas frente a intrusiones que se integrarán en un AIRS (Automatic Intrusion Response System) basado en ontologías. La arquitectura propuesta consta de seis componentes: Sistemas de Detección de Intrusiones, Razonador AIRS, módulo central de ejecución, módulo de comunicación, agentes de ejecución y los componentes de seguridad. El agente de ejecución se basa en plugins, por lo que permite un fácil despliegue de nuevas acciones de respuesta que interactúan con otros componentes de seguridad. Para validar la arquitectura propuesta se ha desplegado una red de prueba con VNX (Virtual Network over linuX), en el que se ejecutan diferentes ataques procedentes de dentro y fuera de la red de la organización obteniendo resultados satisfactorios.*

Palabras clave: *Seguridad en Redes, Sistemas de Detección de Intrusiones, Sistemas de Respuesta a Intrusiones, Snort, SnortSam.*

Abstract: *This paper proposes a distributed, secure and scalable architecture to implement response actions that will be integrated into Ontologies-based AIRS. The proposed architecture involves six components: Intrusion Detection Systems, AIRS Reasoner, Execution Central Module, Communication Module, Execution Agents and the Security Component. The Execution Agent is based on plugins, so it allows easy deployment of new response actions that interact with other Security Components. To validate the proposed architecture we have deployed a test network using the Virtual Network over LinuX tool. We deploy different attacks originating from inside and outside the organization network obtaining satisfactory results.*

Keywords: *Network Security, Intrusion Detection Systems, Intrusion Response Systems, Snort, SnortSam*

1. INTRODUCCION

Al día de hoy, sin lugar a duda hemos sido testigos de la espectacular evolución de las tecnologías de la información: las redes de telecomunicaciones, los terminales y una infinidad de aplicaciones y servicios han dado lugar a una sociedad cada vez más conectada. La mayor parte de la población cuenta con al menos un ordenador en casa o en la oficina, e incluso con teléfonos inteligentes, desde los cuales se accede a varias aplicaciones y servicios sobre la Internet. Así mismo, los sectores: industrial, bancario, militar, educativo, gubernamental y comercial; por mencionar algunos, ofertan cada vez más servicios sobre la Internet de tal forma que sus usuarios pueden acceder a ellos desde cualquier lugar del mundo. Sin embargo, como se puede constatar en el Reporte de Amenazas de Seguridad de Internet del año 2012 emitido por Symantec [1], a la par de ésta evolución, Internet se ha convertido en el sitio preferido de crackers que constantemente y de forma creciente buscan vulnerabilidades (4.989 nuevas vulnerabilidades en el 2011) y las explotan utilizando herramientas y técnicas cada vez más sofisticadas y fáciles de usar; herramientas que permiten a los crackers crear nuevo malware y montar un ataque completo sin tener que escribirlo desde cero. En cualquier caso, las consecuencias de un ataque se traducen en un atentando contra la integridad, confidencialidad y disponibilidad de la información, ya sea que esté almacenada y sean accesibles desde Internet o que transite a través de ella.

En este sentido, el campo de la seguridad en redes ha requerido de una constante investigación para evaluar y optimizar los mecanismos de control de acceso y protección de la información en tránsito, que permitan mitigar los ataques. Tradicionalmente, se han empleado los firewalls y routers para aplicar una política de seguridad y definir listas de control de acceso respectivamente; sin embargo, estos dispositivos de defensa perimetral no pueden garantizar una protección del 100% contra los ataques existentes. Es así que

surgen los Sistemas de Detección de Intrusiones (IDSs), como un nuevo mecanismo de defensa que permite detectar intrusiones o intentos de intrusiones que atenten contra la integridad, confidencialidad y disponibilidad de un recurso.

Los IDSs no son una tecnología nueva, el primer trabajo acerca de estos sistemas fue realizado por James Anderson en 1980 [2]; no obstante, esta tecnología desde sus inicios ha sido sujeta a una constante revisión y evolución. Es así que los Sistemas de Detección de Intrusiones, como menciona Anuar et al. en [3], además de ejecutar técnicas de detección llevan a cabo varias acciones de respuesta en contra de ellas, dando lugar a los Sistemas de Prevención de Intrusiones (IPS) y a los Sistemas de Respuesta a Intrusiones (IRS).

Los tres sistemas tienen por objetivo monitorizar y detectar comportamientos anómalos suscitados en un sistema computacional y/o en la actividad de una red, sin embargo se diferencian en la forma en cómo reaccionan o responden ante un incidente. Un IDS es capaz de ejecutar el proceso de detección y tradicionalmente genera una advertencia o alerta hacia el administrador del sistema, para que sea él quien lleve a cabo una respuesta. El IPS por su parte, lleva a cabo el mismo proceso de detección que el IDS, pero su respuesta no se limita a una simple alerta, sino que es capaz de ejecutar una acción proactiva para evitar que un ataque cumpla su objetivo, la respuesta normalmente consiste en acciones que bloquean el tráfico y lo llevan a cabo a través de un firewall o un dispositivo de control de acceso que debe estar ubicado en línea respecto al tráfico que se intenta filtrar [4]. El IRS también ejecuta una acción de respuesta, pero a diferencia del IPS sus acciones de respuesta no son ejecutadas necesariamente sobre dispositivos en línea con el flujo del tráfico y su acción es reactiva, es decir, si puede prevenir un ataque lo hará, pero su tarea más bien es reducir los daños causados por una intrusión. El IRS tiene por objetivo proveer un catálogo con varias acciones de respuesta, y la elección de una de ellas dependerá de un análisis previo que valore el costo que supone una intrusión respecto del costo de ejecutar una acción de respuesta, de este modo se evita que las respuestas causen mayor daño que las propias intrusiones y además se minimiza el impacto del incidente [5].

En virtud de lo antes mencionado, se pueden identificar dos líneas de actuación claras, y que son sujetos de investigación actual: por un lado están las técnicas de detección de intrusiones, una revisión del estado del arte de estas técnicas se presenta en [6], [7] y [8]; y por otro están las técnicas de respuesta a intrusiones, línea en la cual se incluye el presente trabajo.

Los IRS han sido objeto de varias investigaciones desde hace algunos años: Stakhanova et al., presenta una taxonomía de los IRS junto con una revisión de las tendencias de investigación en la respuesta a intrusiones [9]; Anuar et al., hace un análisis y comparación entre 34 diferentes productos comerciales y no comerciales haciendo distinción entre las diferentes opciones de respuesta [10], y; Shameli-Sendi et al., presenta una revisión de los IRS clasificándolos de acuerdo a una taxonomía, en la que entre otros criterios se considera el método de selección de respuesta, la capacidad de adaptabilidad, el modelo de costo de respuesta y el tiempo de respuesta para clasificarlos [11].
Una de las investigaciones que van en la línea de los IRS que son capaces de adaptar la decisión de respuesta de forma automática en función del contexto y el costo que implica ejecutarlo, es el Sistema Autónomo de Respuesta a Intrusiones basado en Ontologías [12] [13]. Mateos et al. propone la arquitectura de un AIRS basado en ontologías, usando lenguajes formales de especificación de comportamiento y mecanismos de razonamiento, con el fin de garantizar coherencia semántica en un entorno heterogéneo, de tal forma que el AIRS tenga la capacidad de entender la sintaxis y semántica de las alertas de intrusiones generadas por diferentes IDS. El objetivo de la arquitectura propuesta es ejecutar una respuesta óptima luego de evaluar ciertas métricas, como la severidad de una alerta, importancia de un host, etc., además de identificar si dos o más alertas se refieren a una misma intrusión o son diferentes.

Precisamente dentro de este contexto se enmarca el presente artículo. El objetivo es contribuir al AIRS basado en ontologías, mediante una propuesta de arquitectura y posterior implementación del módulo de ejecución de respuestas así como también con la implementación de pruebas de concepto de algunas acciones que serán agregadas al *toolkit* de respuestas para validar al AIRS basado en ontologías.

El resto del artículo está organizado como sigue. Una revisión del Sistema Autónomo de Respuesta a Intrusiones Basado en Ontologías se realiza en la sección 2. La sección 3 presenta la arquitectura distribuida propuesta. La validación de la arquitectura a través de un escenario de experimentación se efectúa en la sección 4. Finalmente, las secciones 5 y 6 presentan los posibles trabajos futuros y conclusiones respectivamente.

2. SISTEMA AUTÓNOMO DE RESPUESTA A INTRUSIONES BASADO EN ONTOLOGÍAS (AIRS)

El AIRS es un sistema de respuesta automática a intrusiones, cuyo principal aporte consiste en la adición de la coherencia semántica al conjunto de características que se espera cumpla un IRS. Aquello es imprescindible, ya que en un entorno de red heterogéneo, donde intervienen diferentes IDSs, cada uno maneja alertas en diferentes formatos y diferente sintaxis [12], [13].

En la Fig. 1, se presenta la arquitectura del AIRS basado en ontologías. El objetivo del AIRS es escoger la respuesta óptima de entre un conjunto de respuestas recomendadas; para ello hace uso de una ontología que es representada mediante un lenguaje formal de representación de conocimiento. Las clases que conforman la ontología son

utilizadas para representar ciertas políticas que son definidas por el administrador, y en base a los cuales se realiza el proceso de inferencia. A continuación se describe cada uno de los módulos que conforman la arquitectura.

Receptor de alertas: Recepta las alertas de intrusiones provenientes de diferentes IDSs con diferentes formatos y sintaxis, y posteriormente mapea los campos incluidos en estas alertas a sus conceptos equivalentes definidos en la ontología.

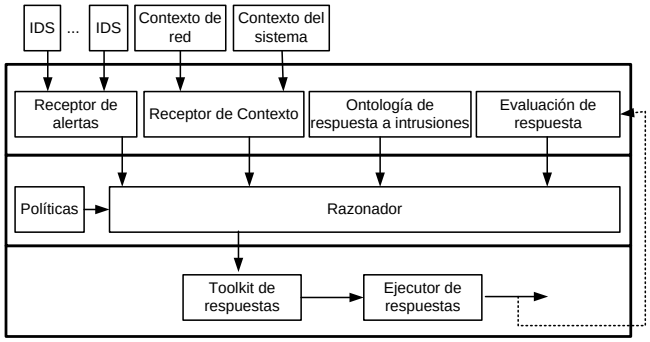

Figura 1. Arquitectura del AIRS basado en Ontologías [10]

Contexto de red: El objetivo es calcular un *parámetro de anomalía de red*, en base a la diferencia existente entre dos snapshots del tráfico de red requerido: el primero, que es obtenido en condiciones normales de operación, y; el segundo, que es obtenido en tiempo real cuando una intrusión es detectada.

Contexto del sistema: El objetivo es calcular un parámetro de anomalía del sistema, en base a la diferencia entre dos estados que evalúan ciertos parámetros del sistema antes y después del ataque: el número de procesos activos, número de procesos zombies, porcentaje de uso de memoria y espacio en disco son algunos ejemplos de ellos.

Receptor de contexto: Recepta la información de contexto de red y del sistema y los mapea a sus correspondientes conceptos definidos en la ontología.

Ontología de respuesta a intrusiones: Define formalmente toda la información del dominio de respuestas a intrusiones necesaria para inferir la respuesta óptima ante una intrusión. Dicho dominio, como se puede constatar en la Fig. 2, está conformado por todas las entidades necesarias que definen el entorno del problema en cuestión: la red propiamente dicha, componentes del sistema, sistemas de detección de intrusiones, sistemas de respuesta a intrusiones, alertas de intrusiones, contexto de red, contexto del sistema, respuestas y resultado de una respuesta. Cada una de estas entidades es representada mediante clases, propiedades y sus relaciones, utilizando el lenguaje estándar OWL (Ontology Web Language) definida por la W3C [14].

Políticas: Están compuestas por un conjunto de reglas que definen el comportamiento del AIRS. Estas reglas son definidas por el administrador del sistema utilizando el lenguaje SWRL (*Semantic Web Rules Language*) [15] y conforman las denominadas métricas de respuesta. La definición de estas métricas y el algoritmo para selección de la respuesta óptima es presentada por Mateos et al. en [12].

Razonador: Es el componente principal del AIRS y es el encargado de inferir la respuesta óptima para una intrusión dada. Para ello toma como entrada las políticas definidas por el administrador y una instancia de la ontología antes mencionada.

Toolkit de respuestas: Es el conjunto de acciones de respuesta disponibles que pueden ser inferidas por el AIRS; dichas respuestas pueden ser activas y pasivas.

Ejecutor de respuestas: Lo constituyen los componentes de seguridad que intervienen en la ejecución de una acción de respuesta.

Evaluación de respuesta: Evalúa si una respuesta previa fue satisfactoria. El resultado es almacenado en una BDD y el histórico es utilizado para futuras respuestas.

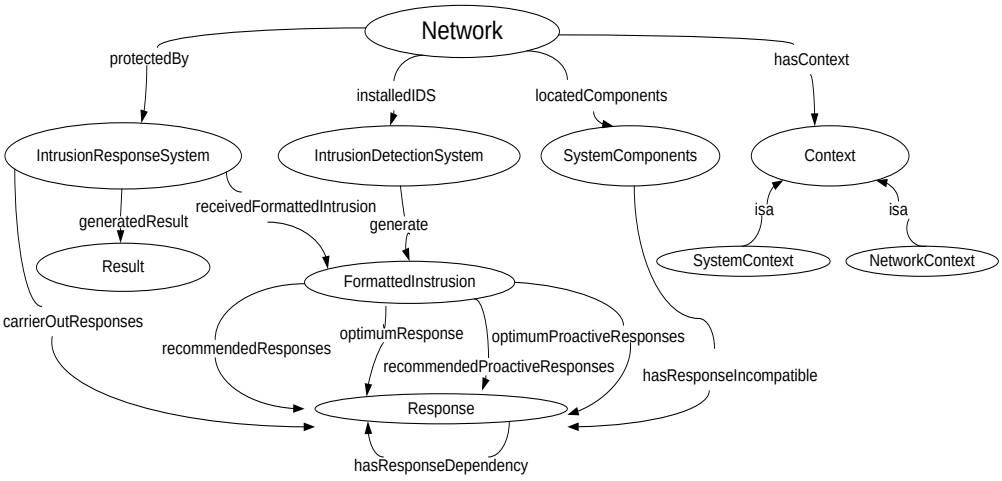

Figura 2. Ontología para la Ejecución Automática a Intrusiones

3. PROPUESTA DE ARQUITECTURA PARA LA EJECUCIÓN DE RESPUESTAS

En la Fig. 3 se realiza una aproximación de la arquitectura distribuida del ejecutor de respuestas, en donde se identifican los siguientes componentes: los IDSs, el razonador que forma parte del AIRS basado en ontologías, el módulo central de ejecución, módulo de comunicación, los agentes de ejecución y los componentes de seguridad.

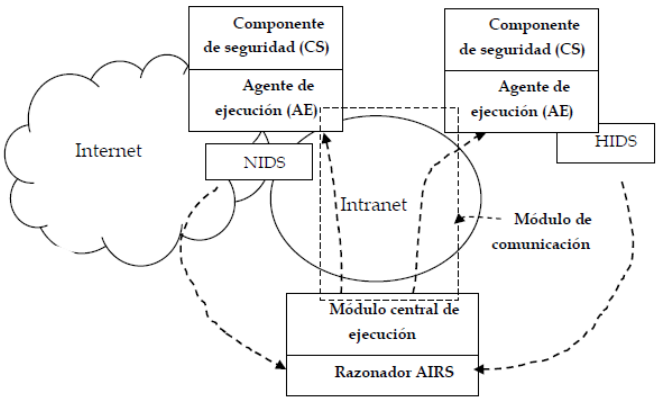

Figura 3. Organización de la Arquitectura Distribuida del Ejecutor de Respuestas

5.1 Descomposición modular de la arquitectura propuesta

El estilo de descomposición modular está orientado a una agrupación de funciones afines, en donde cada módulo procesa los datos de entrada y proporciona los datos de salida respectivos. Cada módulo funciona de manera independiente, y para su comunicación basta con conocer las interfaces de entrada y salida respectivas.

Como se puede apreciar en la Fig. 4, la arquitectura del ejecutor de respuestas según su descomposición modular no difiere en gran medida con la organización del sistema establecida en la Fig. 3. La única diferencia radica en la inclusión del módulo gestor de plugins, adoptado de la arquitectura de SnortSam, una herramienta *Open Source* que ha sido modificada y adaptada para nuestros propósitos.

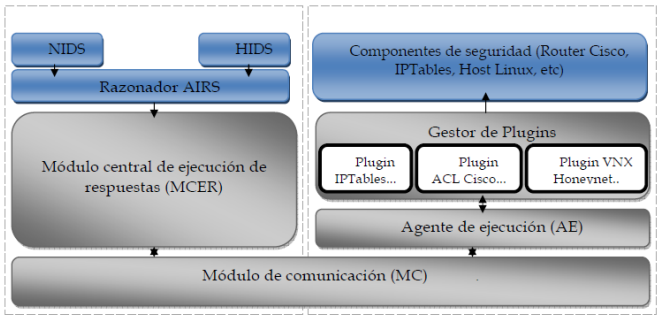

Figura 4. Descomposición Modular del Ejecutor de Respuestas

Ahora bien, para que el ejecutor de respuestas pueda trabajar como un sistema completo cada uno de los módulos deben ser controlados de tal forma que sus funciones se lleven a cabo en el momento adecuado, de acuerdo a una secuencia específica. Por consiguiente, en la Fig. 5 se presenta un diagrama de actividades que define un estilo de control genérico empleado por el ejecutor de respuestas.

IDSs

Pueden ser IDSs basados en hosts (HIDS) o IDSs basados en red (NIDS), quienes tras detectar una intrusión notifican mediante una alerta al AIRS basado en ontologías.

Los IDSs, dentro de la arquitectura propuesta, actúan como sistemas externos y son quienes inician el proceso, más no son los que deciden ejecutar una respuesta. La razón es que las alertas emitidas son analizadas por el *AIRS basado en ontologías* y luego de un proceso de inferencia en base a ciertas métricas de respuesta se elige la respuesta óptima.

Razonador AIRS

Una alerta de intrusión es recibida por medio del receptor de alertas del AIRS basado en ontologías, quien se encarga del *parsing* de las alertas a una instancia de alerta de la ontología del AIRS. Posteriormente se realiza el proceso de inferencia en base a las métricas de respuesta, seleccionando la respuesta óptima a ejecutarse. Una vez que el razonador AIRS ha inferido la respuesta óptima, invoca dicha respuesta. En dicha invocación se envían los parámetros requeridos por una acción de respuesta específica que son obtenidos de la alerta de intrusión o utilizando alguna herramienta automatizada.

Es importante señalar que el razonador AIRS es quien debe garantizar y proporcionar los parámetros necesarios relacionados al ataque que permitan ejecutar una acción de respuesta. Por ejemplo:

- Una acción de respuesta con nombre *DenegarConexion*, que bloquea el tráfico asociado a una conexión específica sobre un firewall, debe invocar la acción de respuesta proporcionando las direcciones IP origen y destino, puertos y protocolo empleado en la conexión.
- Por su parte, una acción de respuesta con nombre *DeshabilitarUsuario*, que deshabilita a un usuario que intenta perpetrar un ataque, debe invocar la acción de respuesta proporcionando la dirección IP del host atacado y el nombre de usuario.

Los parámetros pueden ser obtenidos desde las alertas o utilizando otras herramientas como *nagios* y *net-snmp*. En cualquier caso, el módulo central de ejecución, que se describe en la siguiente sección, asume que los parámetros relacionados a la intrusión serán suministrados por el razonador AIRS.

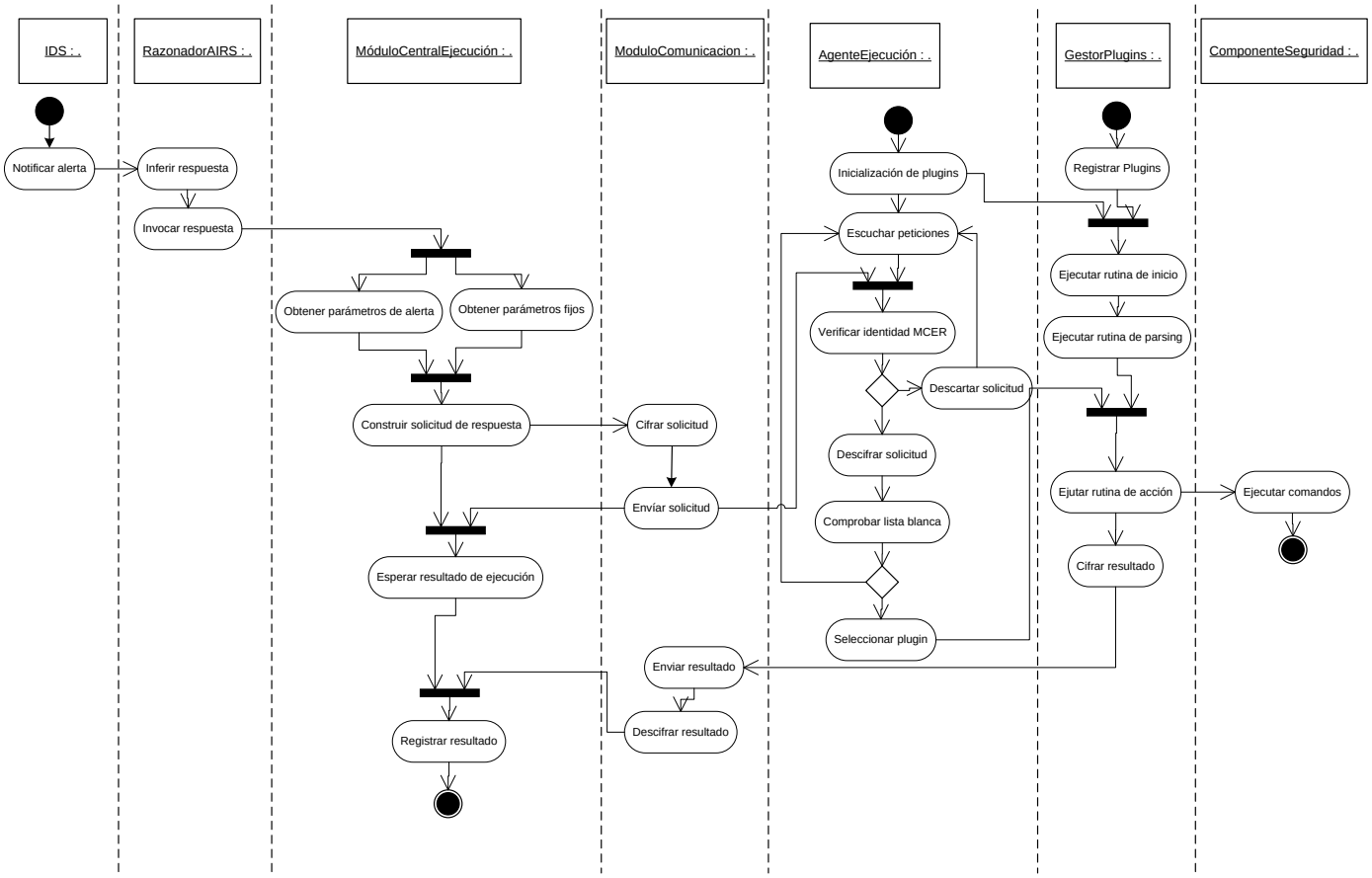

Figura 5.Esquema de Control Genérico del Ejecutor de Respuestas.

Módulo Central de Ejecución (MCER)

El módulo central de ejecución de respuesta se encarga básicamente de dos tareas: construir una solicitud de respuesta e identificar el o los agentes de ejecución a donde se enviará la solicitud de respuesta. En la Fig. 6 se observa el modelo de datos del MCER. Para construir una solicitud de acción de respuesta, el MCER recibe los parámetros de dos fuentes:

a) Desde el razonador AIRS, quien envía los parámetros obtenidos desde las alertas u obtenidos mediante herramientas adicionales. (*ResponseActionsParams* de la Fig. 6).

b) De información fija proporcionada por el administrador, que incluye:

- Grupo de agentes de ejecución, hacia donde se envía una solicitud de respuesta. Cada agente de ejecución contiene información para su localización: dirección IP, puerto y contraseña de cifrado. (*ExecutorAgent* y *GroupExecutorAgents* de la Fig. 6).
- Grupo de Identificadores de alerta (*Signatures ID*), que determina ante qué intrusiones, identificadas por su SID, es posible ejecutar una acción de respuesta. (*SidsGroup* de la Fig. 6).
- Identificador de plugin, asegura que sobre el otro extremo de la comunicación, es decir sobre el agente de ejecución, se ejecute únicamente el plugin que coincida con este valor. (Plugin de la Fig. 6).
- Duración de la respuesta (*duration*), determina el intervalo de tiempo que tendrá efecto una acción de respuesta.
- Modo de ejecución (*mode*), determina el criterio en base al cual se ejecuta una acción de respuesta basada en red. En función del tráfico de entrada (*in*), salida (*out*), entrada y salida (*inout*) o de esa conexión específica (*conn*). Esta opción se hereda de SnortSam.
- Identificación del atacante (*who*), indica qué dirección IP de la alerta es considerado el atacante: origen o destino. Evita ejecutar equivocadamente una acción de respuesta y ocasionar una denegación de servicio en nuestra red.
- Identificación de respuesta compuesta (*composed*), indica si la respuesta está constituida por varias acciones.
- Respuestas (*responses*), solo es utilizada si el campo anterior es verdadero y hace referencia a las respuestas simples que conforman la respuesta compuesta.

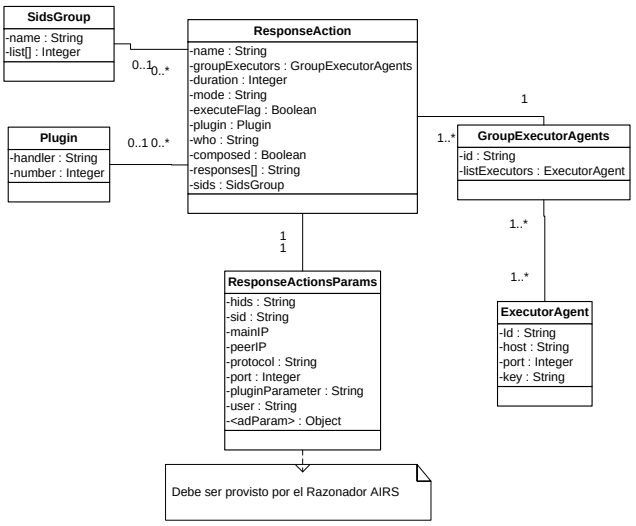

Figura 6.Modelo de Datos del Módulo Central de Ejecución de Respuestas

Módulo de Comunicación (MC)

El objetivo principal de este módulo es proveer un marco común de comunicación entre el MCER y los agentes de ejecución. Independientemente del tipo de acción de respuesta a ejecutar sobre el agente de ejecución éste módulo debe proveer servicios de entrega confiable y segura de las solicitudes de acción de respuestas activas y pasivas inferidas por el razonador AIRS y construidas por el MCER.

- Comunicación confiable: Puesto que la comunicación entre el MCER y el agente de ejecución debe ser confiable, el establecimiento de conexión se efectúa por medio de protocolo de transporte TCP. El agente de ejecución escucha a través de un puerto que es conocido de antemano por el MCER.
- Comunicación cifrada: Las solicitudes de acción de respuesta que viajan a través de la red se cifran utilizando el algoritmo de cifrado en bloques simétrico.

El marco común de comunicación permite el despliegue de varias acciones de respuesta sobre componentes de seguridad remotos en los cuales se ha instalado un agente de ejecución.

Agente de Ejecución

El agente de ejecución actúa como un servicio que siempre se está ejecutando a la espera de la llegada de nuevos paquetes de solicitud de respuesta. Ante el arribo de un paquete de solicitud, selecciona el plugin adecuado y ejecuta el conjunto de comandos asociados a una respuesta activa o pasiva sobre un componente de seguridad. La primera tarea que lleva a cabo el agente de ejecución es la inicialización de plugins, los mismos que han sido registrados por el gestor de plugins y que se tratan en mayor detalle en la siguiente sección.

Luego el agente de ejecución entra en un proceso de espera de paquetes de solicitud de respuesta. Cuando el agente de ejecución recibe un paquete de solicitud de respuesta a través del módulo de comunicación, primero verifica la autenticidad del emisor de la solicitud de acción de respuesta. Si el emisor es auténtico, entonces se descifra el mensaje y se verifica que la solicitud no esté dirigida hacia uno de los componentes que se encuentra en lista blanca.

Finalmente el agente de ejecución selecciona el plugin especificado en el paquete de solicitud y llama a la rutina respectiva para la ejecución de la acción de respuesta sobre el componente de seguridad respectivo. Al término de la ejecución el agente envía de vuelta un resultado que es almacenado en un archivo de *logs* del MCER.

Otra de las tareas que lleva a cabo el agente de ejecución es el control del tiempo que tendrá efecto una acción de respuesta, tal como se mencionó anteriormente, el MCER define el valor de tiempo que tendrá efecto una respuesta. Este valor es almacenado por el agente de ejecución y sirve para deshacer una acción de respuesta.

Gestor de Plugins

El gestor de plugins es un componente importante dentro de la arquitectura del ejecutor de respuestas propuesto, ya que permite resolver el problema de la heterogeneidad de los componentes de seguridad con los que tiene que interactuar el agente de ejecución. Una solicitud de respuesta siempre tiene la misma estructura, independientemente del componente de seguridad; no obstante, cada componente de seguridad tiene su propia sintaxis e interfaz de línea de comandos, y requiere parámetros específicos para su ejecución. Mediante el encapsulamiento de la lógica de control de una acción de respuesta en un plugin, el ejecutor de respuestas puede ser utilizado como una plataforma escalable para desplegar nuevas acciones de respuesta sobre diversos componentes de seguridad sin tener la necesidad de modificar otros módulos del ejecutor de respuestas.

El gestor de plugins permite el manejo de varios componentes de seguridad a través del despliegue de plugins, con el objetivo de resolver algunos problemas que incluyen básicamente 3 aspectos:

- Cada componente de seguridad requiere diferentes parámetros de configuración: iptables, requiere de la interfaz de red sobre la que se aplicará una regla; honeynet VNX, requiere el nombre de usuario y la ruta del archivo VNX; router cisco, requiere dirección IP y contraseña. Entonces el plugin tiene que separar cualquier dependencia de la dotación de los parámetros del agente de ejecución.
- Cada componente de seguridad tiene sus propios comandos de configuración y requiere una lógica de control específica de dicho componente; por consiguiente, el plugin tiene que encapsular la lógica de control.
- Cada componente de seguridad tiene diferentes capacidades en cuanto a la posibilidad de deshacer una acción, soportar múltiples hilos de ejecución, requerir una función de *keepalive*, entre otros. Esta información es requerida por el agente de ejecución y por tanto debe ser proporcionada en el registro del plugin.

Como se observa en la Fig. 7, el gestor de plugins provee dos interfaces: una interfaz de registro de plugins y una interfaz de consulta de plugins.

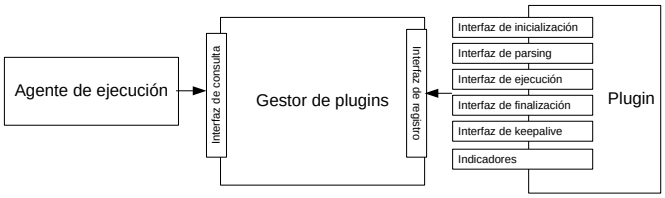

Figura 7. Interfaces del Gestor de Plugins

La interfaz de registro permite el registro de un nuevo plugin, para lo cual el plugin ha de proveer de cinco interfaces y ciertos indicadores necesarios para la operación del agente de ejecución.

- Interfaz de inicialización: Utilizada para ejecutar una rutina de inicio que verifique ciertas precondiciones requeridas para ejecutar una respuesta. Si la rutina de inicio no se ejecuta correctamente, entonces el plugin es deshabilitado. Por ejemplo, la respuesta activa de recuperación que restaura los archivos de un sitio web, requiere de un cliente FTP que se conecta al servidor para descargar ciertos archivos. Entonces se puede comprobar previamente que el agente de ejecución cuente con un cliente FTP para acceder al servidor, caso contrario el plugin es deshabilitado.
- Interfaz de *parsing*: Utilizada para ejecutar una rutina de *parsing* de los parámetros de configuración requeridos para ejecutar una respuesta sobre un componente de seguridad. Por ejemplo, la respuesta activa de protección que agrega una lista de control de acceso a un router perimetral Cisco, requiere la dirección IP del router y las contraseñas respectivas.
- Interfaz de ejecución: Utilizada para hacer referencia a la rutina que contiene la lógica de control que lleva a cabo la acción de respuesta. Por ejemplo, la respuesta activa de decepción que despliega una red señuelo usando VNX, incluirá todos los comandos propios de VNX que permitan la ejecución de dicha respuesta.
- Interfaz de finalización: Utilizado para ejecutar una rutina de salida en el momento en que el ejecutor de respuestas se cierra. Esto le otorga al plugin la posibilidad de limpiar o ejecutar acciones a conveniencia.
- Interfaz de *keepalive*: Utilizada para ejecutar una rutina de *keep-alive*, cuya función es mantener conexiones persistentes con componentes de seguridad que así lo requieran. Por ejemplo puede ser necesario mantener una conexión con un firewall perimetral para evitar realizar *logins* frecuentes.
- Indicadores: Varios indicadores pueden ser requeridos durante la fase de registro del plugin y pueden ser añadidos conforme a las necesidades del agente de ejecución. Por ejemplo una función del agente de ejecución es deshacer una acción de respuesta después de un tiempo determinado; no obstante, algunas respuestas como por ejemplo la restauración de ficheros, no requieren esta función; por lo tanto a través de un indicador se dice al agente de ejecución que dicho plugin no requiere esa funcionalidad.

Componentes de seguridad

Representa el dispositivo que lleva a cabo la acción de respuesta real utilizando su interfaz de línea de comandos, tal que se altere su funcionamiento tan pronto como es ejecutado. Un router, firewall, servidor web, servidor FTP, gestor de usuarios, gestor de procesos son algunos ejemplos de ellos.

4. ESCENARIO DE EXPERIMENTACIÓN

La validación de la arquitectura del ejecutor de respuestas se llevó a cabo en una red ad hoc que se muestra en la Fig. 8. En lo posible se ha intentado emular una topología de red que comúnmente es desplegada en las organizaciones; por consiguiente, se han definido varias subredes a la que se conectan hosts con diferentes sistemas operativos. La red está constituida de los componentes que se muestran en la Tabla 1.

Tabla 1. Componentes empleados en la Red de Prueba

Componente	Características
RFW y RINT	Router Cisco C3640 Version 12.3(26)
ATT (Atacante)	GNU/Linux Ubuntu 10.04.3 LTS con BackTrack 5
INT1-1, INT1-3, INT2-1 e INT2-5	GNU/Linux Ubuntu 11.04
INT1-2, INT1-4, INT2-2, INT2-3, INT2-4 e INT2-6	Windows XP Service Pack 3
INT3-1 e INT3-2	GNU/Linux Ubuntu 11.04 HIDS OSSEC versión 2.7
DMZ-1	GNU/Linux Ubuntu 8.0.4 con Metasploitable
DMZ-2	GNU/Linux Ubuntu 11.04
IDS-1, IDS-2 e IDS-3	GNU/Linux Ubuntu 10.04.2 TTS NIDS Snort versión 2.9.2.3
AIRS basado en ontologías.	GNU/Linux Ubuntu 12.04.1 LTS Ordenador: Dell XPS L502X, 8GB RAM, Procesador 2.6GHz.

La topología de la red ha sido creada e implementada utilizando la herramienta de virtualización VNX (Virtual Network over LinuX).

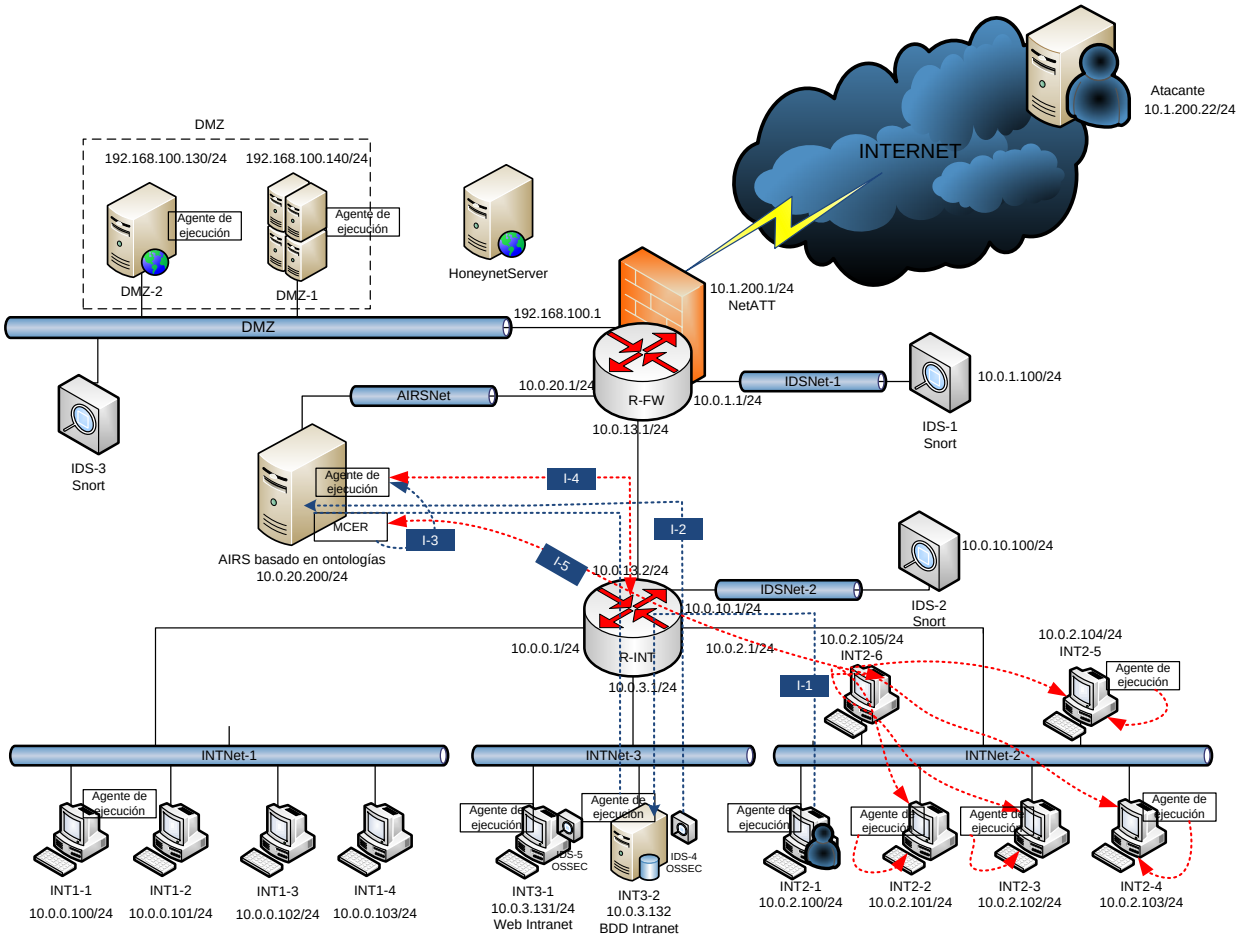

Figura 8. Ataque y Respuesta desde un Host Interno hacia el Servidor de BDD de la Intranet.

Con el propósito de mostrar los resultados obtenidos y la interacción entre los módulos propuestos e implementados, se divide el proceso en 3 actividades: detección, inferencia y ejecución.

Detección

I-1: El atacante (10.0.2.100) ubicado en la red interna INTNet-2, intenta establecer una conexión ssh, por medio de un ataque de fuerza bruta, hacia el servidor de BDD de la intranet (10.0.3.132).

I-2: Al ser el servidor de BDD un host crítico, se ha instalado el HIDS OSSEC. OSSEC realiza la verificación de integridad de ficheros, monitorización y análisis de *logs* del sistema, y detección de *rootkits*. En virtud de ello, al detectar el ataque emite una alerta de intrusión en formato *syslog* hacia el razonador AIRS. El razonador AIRS, escucha a través de su módulo de recepción de alertas en el puerto 512 y recibe las alertas en formato *syslog*.

Inferencia

Luego de la recepción de la alerta, el razonador AIRS infiere la respuesta óptima. En este caso el razonador infiere una respuesta activa de protección, de nombre *isolateHost*, que tiene por objetivo aislar al host atacante. La respuesta inferida está compuesta por dos acciones simples: 1) la acción *addACL*, que agrega una regla a la lista de control de acceso del router interno (R-INT), y; 2) la acción *blockInINTNet*, que agrega una regla de filtrado al firewall personal de cada host que pertenece a la misma subred del atacante. Para la ejecución de la respuesta llama al módulo central de ejecución de respuestas (MCER), a quien provee el nombre de respuesta inferida (*isolateHost*), y los parámetros requeridos; en este caso únicamente la dirección IP del atacante (10.0.2.100).

Ejecución

I-3: El MCER recibe la solicitud de respuesta desde el razonador AIRS y recupera toda la información necesaria para ejecutar la respuesta *isolateHost*. Dicha información señala que la respuesta *isolateHost* está compuesta por dos acciones: *addACL y blockInINTNet2*.

- La acción *addACL*, se ejecuta sobre el agente de ejecución local. Dicho agente interactúa con un router Cisco por medio del *plugin cisoacl*, añadiendo una regla para el filtrado del tráfico del atacante.

- La acción *blockInINTNet2*, se ejecuta sobre los agentes de ejecución de los 5 hosts pertenecientes a la subred del atacante (INTNet-2). Cada agente interactúa con el el firewall IPTables por medio del *plugin ipt-block*, añadiendo una regla de filtrado del tráfico del atacante.

Debido a que la respuesta a ejecutar involucra a seis agentes de ejecución, el MCER a través del módulo de comunicación debe establecer seis conexiones y emitir las solicitudes de respuesta correspondientes.

Previo a la ejecución de la acción de respuesta (en la fase inicialización) ambos plugins involucrados, parsean los parámetros necesarios para su ejecución. El plugin *ciscoacl* obtiene: la dirección IP del router, dirección IP del servidor TFTP, las contraseñas necesarias, el nombre de la ACL, el tipo de ACL y la interfaz en la que se aplicará. Por su parte, el plugin *ipt-block* parsea la interfaz sobre la cual se agrega la regla de filtrado.

I-4: El agente de ejecución local a través del plugin *ciscoacl* se comunica con el router R-INT (10.0.2.1) y realiza las siguientes tareas:

- El momento de llamar la función de parsing, descarga el fichero runnig-config al servidor TFTP.
- Sobre dicho fichero se busca si se ha definido antes una ACL (Lista de Control de Acceso) con el mismo nombre que el definido como parámetro (AIRS-ACL en este caso). Si ya existe una ACL definida, entonces se agrega una nueva en el lugar adecuado de dicha lista. Si no existe la ACL, entonces se crea una nueva ACL al fichero.
- Se establece una conexión con el router y posteriormente se envían los datos para la autenticación.
- Finalmente se sube el nuevo fichero running-config hacia el router desde el servidor FTP.

Nueve escenarios distintos, similares a las presentadas en la sección anterior, han sido implementados para la validación de la arquitectura propuesta, obteniendo resultados totalmente satisfactorios que serán extendidos en futuras publicaciones.

5. TRABAJOS FUTUROS

Una vez provisto e implementado el ejecutor de respuestas, nuevos plugins pueden ser implementados. Los plugins provistos en este trabajo son pruebas de concepto que pueden ser mejorados a través de la implementación de plugins más "inteligentes", la obtención de ciertos parámetros se pueden obtener de forma dinámica.

La naturaleza basada en plugins, permite definir nuevas funciones que deben ser implementadas por cada plugin. Actualmente, no existe una función que evalúe el efecto que tuvo la acción de respuesta; es decir, si la acción neutralizó o no el ataque. Entonces, una función para efectuar dicha evaluación puede ser muy útil para que el razonador AIRS utilice el resultado de éxito de una acción de respuesta en inferencias futuras.

Actualmente, cuando se requiere ejecutar una respuesta compuesta, es el MCER quien debe ejecutar cada acción simple de forma independiente; no obstante, en determinadas circunstancias podría ser útil que un agente de ejecución tenga la capacidad de ejecutar una acción sobre otro agente de ejecución. Esta funcionalidad podría implementarse como un plugin adicional, para evitar modificar la arquitectura original.

Una arquitectura basada en plugins similar a la propuesta en este trabajo, podría ser extendida hacia los IDSs. Es decir, proveer un marco común de comunicación entre los IDSs y el receptor de alertas, e implementar un plugin que incluya una función para realizar el parsing de cada alerta.

6. CONCLUSIONES

Se presenta la propuesta de una arquitectura para la respuesta automática a intrusiones, que sea distribuida, segura y escalable. Para ello se realizó un análisis de requerimientos, diseño y posterior implementación de un prototipo, que finalmente fue integrado al AIRS basado en ontologías. De la misma manera se ha realizado la implementación de pruebas de concepto de respuestas para el catálogo del AIRS, que aprovechan las características de la arquitectura propuesta.

La utilización de múltiples IDSs distribuidos y de varios tipos (en este caso HIDS y NIDS), permite detectar un rango más amplio de intrusiones. Además que si varios IDSs cubren un mismo conjunto de intrusiones, la confianza sobre las alertas emitidas desde estos IDSs aumenta; por consiguiente se reducen los falsos positivos.

De la misma manera, la provisión de agentes de ejecución distribuidos permite la ejecución de respuestas de forma local y remota, interactuando con diversos componentes de seguridad. Así mismo provee un marco común de comunicación y un conjunto de servicios independientemente de la respuesta a ejecutar. El marco común de comunicación permite establecer conexiones seguras y confiables entre el MCER y los agentes de ejecución. Mientras que la lógica de ejecución de la respuesta, parsing de parámetros de configuración e inicialización, propios de cada componente de seguridad, son provistos a través de plugins.

La respuesta automática que provee el sistema propuesto, permite reducir la ventana de tiempo para actuar ante una intrusión; en lugar de esperar a que el administrador actúe manualmente ante una alerta. Además, a diferencia de otras herramientas como SnortSam, FWSnort o SnortInline, el sistema propuesto no siempre se ejecuta la misma respuesta ante una intrusión determinada; al contrario, se realiza una selección de la respuesta óptima luego de un razonamiento llevado a cabo por un razonador semántico.

REFERENCIAS

[1] Symantec. Symantec Internet Security threat Report, 2011 trends. Symantec Corporation. USA. 2012 Available: http://www.symantec.com/content/en/us/enterprise/other_resources/bistr_main_report_2011_21239364.en-us.pdf.

[2] J. P. Anderson, "Computer Security Threat Monitoring and Surveillance," National Institute of Standards and Technology (NIST), 1980.

[3] N. B. Anuar, M. Papadaki, S. Furnell and N. Clarke, "An Investigation and Survey of Response Options for Intrusion Response Systems (IRSs)," Information Security for South Africa (ISSA), 2010, pp. 1-8, 2010.

[4] M. Papadaki and S. Furnell, "IDS or IPS: what is best?" Network Security, vol. 2004, pp. 15-19, 2004.

[5] N. Stakhanova, S. Basu, J. Wong, D. P. IEEE Tech Comm and Nokia, "A Costsensitive Model for Preemptive Intrusion Response System," pp. 9, 2007.

[6] M. Tavallaee, N. Stakhanova and A. A. Ghorbani, "Toward credible Evaluation of Anomaly-based Intrusion-Detection Methods," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 40, pp. 516-524, 2010.

[7] G. M. Nazer and A. A. L. Selvakumar, "Current Intrusion Detection Techniques in Information Technology–A Detailed Analysis," European Journal of Scientific Research, vol. 65, pp. 611-624, 2011.

[8] S. Mallissery, J. Prabhu and R. Ganiga, "Survey on Intrusion Detection Methods," in Advances in Recent Technologies in Communication and Computing (ARTCom 2011), 3rd International Conference on, 2011, pp. 224-228.

[9] N. Stakhanova, S. Basu and J. Wong, "A Taxonomy of Intrusion Response Systems," International Journal of Information and Computer Security, vol. 1, pp. 169-184, 2007.

[10] N. Anuar, M. Papadaki, S. Furnell and N. Clarke, "An investigation and survey of response options for Intrusion Response Systems (IRSs)," Proceedings of the 9th Annual Information Security South Africa Conference, pp. 1-8, 2010.

[11] A. Shameli-Sendi, Ezzati-jivan N., M. Jabbarifar, and M. Dagenais, "Intrusion Response System: Survey and Taxonomy", International Journal of Computer Science and Network Security, vol. 12, 2012.

[12] V. Mateos, V. Villagrá and F. Romero, "Ontologies-Based Automated Intrusion Response System," Computational Intelligence in Security for Information Systems 2010, pp. 99-106, 2010.

[13] V. Mateos, V. A. Villagrá, F. Romero and J. Berrocal, "Definition of Response Metrics for an Ontology-based Automated Intrusion Response Systems," Comput. Electr. Eng., 2012.

[14] W3C. OWL 2 Web Ontology Language. W3C Recommendation 2012. Available: OWL 2 Web Ontology Language.

[15] W3C. SWRL: A Semantic Web Rule Language. 2004. Available: http://www.w3.org/Submission/SWRL/.