# Arquitectura de Agentes ARSEC-AMS para Diseño de Sistemas de Seguridad

María de Guadalupe Cota Ortiz[a], Juan Pablo Soto Barrera[a], y Pedro Flores Pérez[a]

[a] *Universidad de Sonora.*
*Blvd. Luis Encinas y Rosales S/N, Colonia Centro, Hermosillo, Sonora, México, 83000.*
*{lcota,jpsoto,pflores}@gauss.mat.uson.mx*

*2013 Published by* $\mathcal{DIFU}_{100}ci@$ *http://www2.uaz.edu.mx/web/www/publicaciones*
*Selection and peer-review under responsibility of the Organizing Committee of the CICOMP-2013, www.cicomp.org*

**Resumen**

En este artículo se presenta la arquitectura híbrida para diseño y desarrollo de sistemas de seguridad computacional basados en tecnología de agentes, denominada ARSEC-AMS (Security Architect Multiagent System), la cual permite implementar mecanismos de interacción y razonamiento de agentes para detectar situaciones anormales que pueden representar amenazas no registradas en la base de conocimientos del sistema. El enfoque difiere de otras técnicas y metodologías que se vienen implementando en herramientas de protección, que aunque exitosas, no logran controlar eventos relacionados con el tema denominado "Zero day", problemática que consiste en el surgimiento constante de nuevas formas de intrusión o programas que no son reconocidos por software de protección y toman el control de equipos de cómputo para vulnerar o comprometer sus recursos. La arquitectura cuenta con los módulos intérprete, pizarra, deliberativo-cognitivo, seguridad y reactivo, que en forma coordinada permiten al sistema de agentes alcanzar los objetivos de seguridad previamente establecidos.

*Palabras clave:* Arquitectura, Agente, Seguridad computacional.

## 1. Introducción

La La tecnología de agentes es un campo innovador en el desarrollo de aplicaciones, donde los agentes son identificados como entidades software programadas para realizar una serie de operaciones con cierto grado de independencia, utilizando la información que proviene de su entorno y el conocimiento proporcionado por los diseñadores de este tipo de sistemas para dotarles de formas de razonamiento que les permitan percibir sus capacidades y el alcance funcional para la toma de decisiones [1].

Por otra parte, la evolución tecnológica en materia computacional proporciona a todo tipo de organizaciones y particulares la posibilidad de modernizar sus servicios y aumentar el nivel de ingresos y productividad, pero esto también implica el tener que afrontar problemas de seguridad que cada vez son más complejos y difíciles de resolver, los cuales pueden ocasionar pérdidas millonarias en materia de información o recursos financieros [2]. En la actualidad, los países más desarrollados son los que hacen uso

de este tipo de tecnología, y por lo tanto, son los más susceptibles a delitos cibernéticos que se cometen con el fin de corromper, destruir o modificar los datos que se almacenan en sistemas de cómputo o se transmiten a través de redes de computadoras; situación que los obliga a adoptar medidas de seguridad que permitan proteger las operaciones que se realizan por este medio, y así, poder garantizar un nivel óptimo de confidencialidad, integridad y disponibilidad de la información [3, 4, 5, 6, 7].

Una de las amenazas más graves en seguridad computacional es llamada *amenazas del día cero* o *zero day*, y se identifica como el período de tiempo que inicia cuando se activa *malware* que no puede ser detectado por software de protección como sistemas antivirus, detectores de intrusos, etc., por no tener registrados los patrones de comportamiento en sus bases de datos [6]. Como ejemplo, está el caso de los *gusanos* o *worms*, que generalmente tienen el propósito de comprometer recursos de los equipos que han infectado, exponiéndolos a intrusiones o ataques que aprovechan esta situación, la cual perdura hasta que el problema es detectado y controlado por personal de seguridad [4] en horas, meses, años, o simplemente quedar sin solución por tiempo indeterminado.

Para controlar problemas relacionados con *Amenazas del día cero* o *Zero day* con un enfoque distinto a los actuales, una vez que se revisaron especificaciones de arquitecturas de agentes existentes, se llegó a la conclusión de que, en forma individual, adolecen de elementos básicos para diseñar sistemas de seguridad complejos. Retomando ideas de arquitecturas como BDI [8], GAIA [9], [10] y PRS [11], se ha diseñado la arquitectura híbrida orientada al área de seguridad computacional que ha sido denominada ARSEC-AMS, la cual permite abordar problemas de seguridad con tecnología de agentes, implementar mecanismos de comunicación y razonamiento de agentes, establecer criterios de seguridad y protección de información, aplicar comportamientos reactivos y dotar a los agentes de capacidades para la toma de decisiones e implementación de acciones que repercutan y se reflejen en su entorno. Dicha arquitectura está organizada en los siguientes módulos: el intérprete que se encarga de filtrar los mensajes entrantes y salientes del sistema de agentes, y de la canalización de transacciones que pueden estar dirigidas al control de la pizarra como elemento de comunicación, al módulo deliberativo-cognitivo que permite implementar formas de representación del conocimiento, al módulo de seguridad donde se pueden establecer criterios y

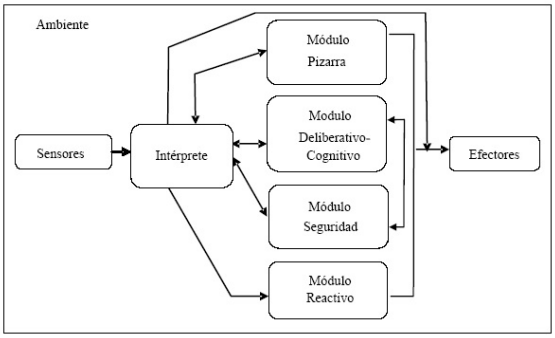


Figura 1. Arquitectura ARSEC-AMS.

niveles de seguridad, registrar alertas, definir instrucciones, etc., o al módulo reactivo que se encarga de implementar acciones programadas que respondan a cierto tipo de eventos previamente definidos por el administrador o por el sistema de agentes.

Cabe mencionar que bajo el modelo de la arquitectura ARSEC-AMS y sus componentes, se han concluido en el último año cuatro tesis en la Licenciatura en Ciencias de la Computación de la Universidad de Sonora, con el desarrollo de sistemas basados en tecnología de agentes para el diseño y desarrollo de herramientas software en materia de seguridad computacional, obteniendo una de ellas mención honorífica.

Para efectos de organización, para el desarrollo de este documento se utilizará el mismo orden en que se han mencionado los módulos de la arquitectura, agregando al final la descripción de elementos complementarios para el diseño y desarrollo de sistemas de seguridad basados en agentes tomando como base la arquitectura ARSEC-AMS.

## 2. Arquitectura de agentes ARSEC-AMS

La arquitectura ARSEC-AMS (ver Figura 1), se compone de módulos que en forma integral, permiten interpretar los eventos del ambiente donde se desenvuelven los agentes, diseñar mecanismos de razonamiento de tipo deliberativo-cognitivo, establecer criterios y niveles de seguridad, implementar tareas de tipo reactivas y mantener el control de la pizarra que se utiliza como elemento de apoyo en el sistema de comunicación.

### 2.1. Roles de agentes

Los roles que han sido diseñados para establecer las distintas funcionalidades de los tipos de agentes tienen

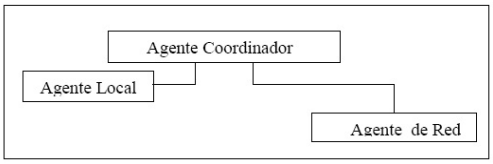


Figura 2. Jerarquía de Agentes.

correspondencia con un nivel jerárquico que define el grado de autoridad, ventajas y limitaciones que pueden asumir los agentes, y se asocian con conceptos relacionados con aspectos de seguridad como lo es el de responsabilidad, permiso y privilegio.

En la jerarquía diseñada para la arquitectura ARSEC-AMS (ver Figura 2), se establecen tres perfiles: Agente Coordinador (AC), Agente Local (AL) y Agente de Red (AR), los cuales se implementan atendiendo los siguientes criterios:

1. Conforme disminuye el nivel jerárquico, el agente tiene menor grado de autoridad para acceder a los recursos del sistema.
2. Cada rol se asocia con un nivel jerárquico, de tal forma que independientemente del grado de jerarquía que tenga un agente, no podrá realizar las funciones de los roles que corresponden a otro nivel.

En este contexto:

a) El Agente AC es el encargado de administrar y coordinar el trabajo especializado del conjunto de agentes y realizar actividades especializadas y complejas.
b) El Agente AL administra el funcionamiento interno del equipo donde se crea y realiza actividades generalizadas para vigilar el cumplimiento de políticas de seguridad, revisar el comportamiento de los recursos del sistema, y monitorear eventos locales en su entorno.
c) El Agente AR supervisa el comportamiento de red del equipo donde se crea, realiza actividades de monitoreo de paquetes de red, detecta eventos que pueden considerarse como anormales e implementa acciones establecidas previamente por el administrador del sistema o por un agente de tipo Coordinador.

## 2.2. Esquema funcional de agentes

Con el fin de llevar un seguimiento de transacciones e intercambio de mensajes del sistema de agentes, los roles han sido diseñados bajo el esquema funcional que se muestra en la Figura 3, estableciendo un control interno para intercambio de mensajes basado en el protocolo *request_protocol* propuesto por FIPA [12] (ver

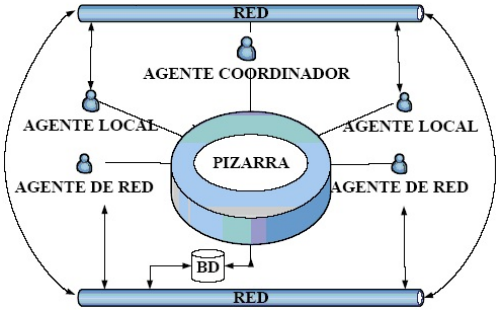


Figura 3. Esquema funcional de agentes.

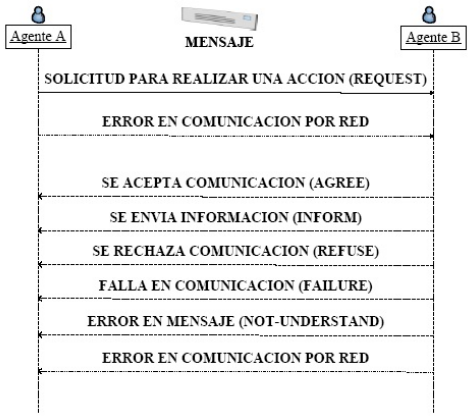


Figura 4. Esquema básico para intercambio de mensajes.

Figura 4). El procedimiento para lograr este objetivo, consiste en establecer un protocolo de comunicación que sirva como base para implementar los mensajes relacionados con las siguientes opciones:

- ***Solicitud para realizar una acción (request)***. Un agente puede solicitar a otro la realización de una acción.
- ***Error en comunicación por red***. Además de los casos contemplados en el protocolo *request_protocol* de FIPA, en esta propuesta se incluye una opción adicional que permite al agente emisor, en base a un tiempo de espera previamente definido, detectar si el mensaje enviado ha llegado a su destino. Cuando se presenta este tipo de error, se recomienda reprogramar la transacción para el envío del mensaje.
- ***Aceptar comunicación (agree)***. Este mensaje, además de informar sobre la aceptación de realizar la acción solicitada, sirve como un aviso de que el mensaje transmitido ha llegado en forma correcta a su destino.
- ***Envío de información (inform)***. Consiste en enviar un mensaje que transmita cierta información sin requerir respuesta.

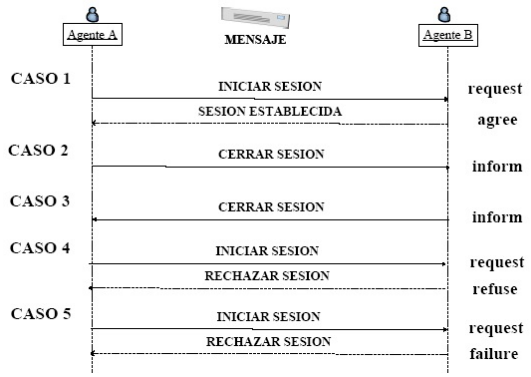


Figura 5. Esquema para control interno de sesiones.

- ***Rechazo de comunicación (refuse)***. Cuando por alguna razón, la solicitud de un agente no puede ser atendida, se envía un mensaje de rechazo, y opcionalmente, se agrega información adicional que indica el motivo por el cual no se puede realizar la acción solicitada o no puede aceptar establecer una sesión de trabajo.

- ***Falla en comunicación (failure)***. Error en secuencia de comunicación. Cuando una comunicación ha sido interrumpida antes de llegar a una confirmación de finalización de la misma por alguna de las partes, pueden emitirse mensajes que informen sobre las fallas de comunicación, y opcionalmente, pueden incluirse reportes con indicios que permitan detectar el origen del problema. En este caso existe la posibilidad de reiniciar completamente la transacción, o continuarla desde el punto donde fue interrumpida.

- ***Error en el mensaje (not-understand)***. Si un mensaje no puede comprenderse pero se tienen los datos de referencia del emisor, puede enviarse un mensaje informado la situación para que se reprograme el envío del mensaje original.

Por otra parte, tomando en cuenta que la arquitectura ARSEC-AMS está orientada a problemas de seguridad computacional, en el caso de control de agentes alojados en múltiples equipos, opcionalmente, para efectos de evitar intrusiones en el sistema de agentes, se recomienda utilizar un esquema de control interno de sesiones de trabajo implementado por agentes AC (ver Figura 5), apoyándose para esta tarea en información interna del sistema de agentes que debe ser centralizada en un sistema remoto de bases de datos.

En base a lo descrito anteriormente, y para efectos de mantener un control interno de las sesiones de agentes y el envío/recepción de mensajes interactivos, se ha diseñado una transacción específica identificada como *sesión*, a través de la cual se implementan los siguientes casos (ver Figura 5):

- ***Iniciar sesión (caso 1)***. Para que un agente pueda solicitar a otro que se realice una acción o se identifiquen como válidos los mensajes recibidos, es necesario que exista una confirmación de sesión establecida, y que este hecho sea registrada en un contenedor de datos, que básicamente contenga el nombre de los agente y el IP o dirección de red. En caso de que una sesión se finalice por cualquier causa, el registro de ésta debe ser eliminado de la lista en forma inmediata.

- ***Sesión establecida (caso 1)***. Cuando un agente recibe una solicitud para iniciar una sesión, puede responder con un mensaje de *Sesión establecida*, con lo cual acepta mantener un canal de comunicación para envío y recepción de mensajes.

- ***Cerrar sesión (Caso 2 y 3)***. Si una transacción ha llegado a su fin o la sesión de trabajo debe finalizarse, el mensaje que debe enviarse es el que corresponde a *Cerrar sesión*.

- ***Rechazar petición de sesión (Caso 4 y 5)***. Si el agente que recibe una solicitud para iniciar una sesión de trabajo, debe rechazar la petición, y opcionalmente, puede agregar información que detalle o explique el motivo de este tipo de respuestas.

- ***Sin respuesta (failure)***. Atendiendo a lo estipulado en el esquema básico, esta es una opción que permite verificar si los mensajes enviados han llegado a su destino o se han presentado errores en la transmisión de los datos y reprogramar la transacción o abortarla.

En el control interno planteado para manejo de sesiones, en los casos 1, 4 y 5 (ver Figura 5), el agente que envía el mensaje *Iniciar sesión*, debe establecer un tiempo límite para obtener respuesta del receptor, de tal forma sea capaz de detectar fallos en el envío/recepción de mensajes, con el fin de reprogramar las peticiones de sesión que no hayan sido atendidas y/o registrar alertas para el administrador que permitan detectar nodos inestables en los equipos donde se encuentra instalado el sistema.

### 2.3. Módulos de la arquitectura ARSEC-AMS

Los comportamientos de agentes de tipo general asociados a los módulos de la arquitectura ARSEC-AMS, pueden presentar variaciones debidas a la identificación

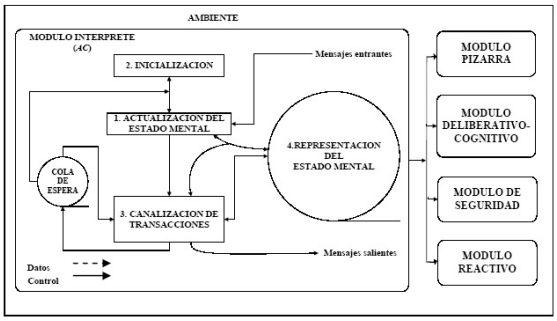


Figura 6. Flujo de mensajes en el módulo intérprete del Agente tipo Coordinador.

de funciones que corresponden al rol y nivel jerárquico del agente que se trate, o por cambios contextuales que ocurren en su medio ambiente.

### 2.3.1. Módulo intérprete

El módulo que funge como *interprete* se encarga de filtrar los eventos registrados en el sistema y dar seguimiento en forma coordinada al conjunto de acciones a realizar para alcanzar los objetivos definidos para el sistema de agentes. Tiene como función interpretar los mensajes relacionados con los eventos filtrados por *sensores* del sistema, o redireccionarlos hacia el sistema de salida a través de *efectores* (ver Figura 6).

Las tareas básicas que sirven de soporte a este módulo para agentes AC son las siguientes:

a) Obtener datos relacionados con los agentes del sistema y coordinar el trabajo conjunto.
b) Crear sesiones de trabajo y mantener control del estado en que se encuentran.
c) Controlar las instancias de pizarra.
d) Aplicar lineamientos o criterios establecidos por el administrador del sistema en la toma de decisiones relacionados con períodos de revisión, instrucciones, políticas, niveles de seguridad, alertas, acciones a tomar, etc.
e) Emitir alertas al administrador cuando se presenten casos en los cuales existe sospecha de un problema de seguridad, y aplicar medidas precautorias para resguardar los recursos involucrados en tanto se definen acciones para el tratamiento de eventos del mismo tipo, o se recibe la instrucción para desactivar en forma permanente la protección preventiva que se haya activado.
f) Mantener un sistema de control en el envío de mensajes para efectos de reprogramarlos en una cola de espera cuando éstos no lleguen con éxito a su destino. Para realizar esta tarea es necesario implementar un mecanismo que permita la detección de la recepción de los mensajes.

La única diferencia en el comportamiento de los agentes en este módulo, es que para los agentes AL y AR el proceso "*3 Canalización de transacciones*" cambia a "*3 Ejecución de instrucciones*".

#### Flujo de control del módulo intérprete

En los puntos marcados como 1 y 2 en la Figura 6 se realizan acciones de inicialización y actualización del estado mental de los agentes para identificar creencias, capacidades y el alcance de funciones de cada rol. En el punto 3 se controla la canalización de transacciones y mensajes salientes, se ejecutan instrucciones, y se mantiene control sobre una lista de espera de transacciones que deben ser reprogramadas. Por otra parte, el punto 4 de la Figura 6 identifica al módulo que proporciona información al sistema de agentes relacionada con la representación del estado mental del agente (creencias), la cual puede ser actualizada con la recolección de datos relacionados con características básicas de agentes, eventos, o información relevante que es extraída de mensajes entrantes, planes a seguir (intenciones), registro de metas u objetivos (deseos), y registrar el estado de avance actual de los objetivos registrados.

### 2.3.2. Módulo Pizarra

Si bien es cierto que uno de los recursos más importantes de un sistema de agentes es la información que se define para efectos de organización y funcionamiento, también lo es la que se genera en forma dinámica durante períodos de actividad, cuyo significado e interpretación proporciona un medio de identificación de patrones en los sucesos que se presentan en el medio ambiente, así como el elemento que se utilice para registrarla. En este sentido, la arquitectura de pizarra sirve como medio de comunicación en un sistema de agentes a través de sus componentes y se constituye en un conjunto de fuentes de conocimiento (KSs, Knowledge Sources), y un mecanismo de control [13].

La arquitectura de pizarra denominada BLACKBOARD-ECRE(Blackboard-Element Control for Register of Events) que se muestra en la Figura 7, ha sido diseñada para implemetar el módulo de pizarra en la arquitectura ARSEC-AMS, para efectos de registrar eventos relevantes que puedan ser analizados con el fin de detectar patrones de comportamiento que puedan constituir un problema de seguridad. Se implementa a través del esquema de base de datos que se muestra en la Figura 8 y funciona como elemento de

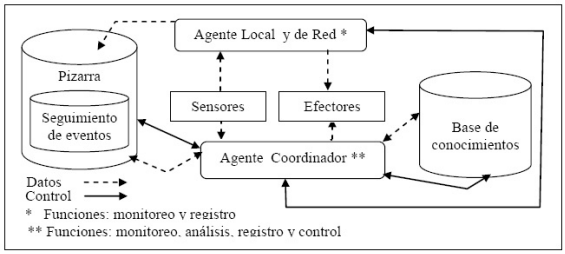


Figura 7. Arquitectura de Pizarra BLACKBOARD-ECRE.

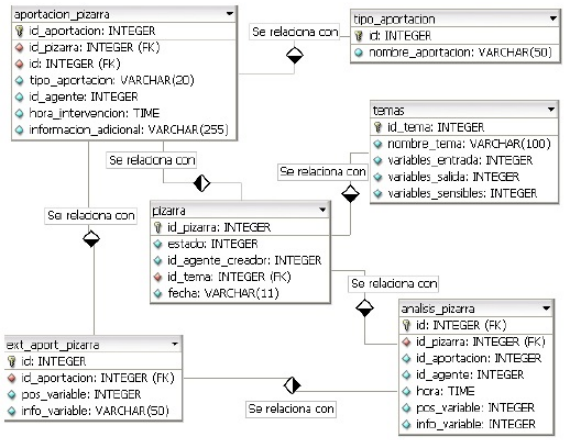


Figura 8. Esquema de Base de Datos.

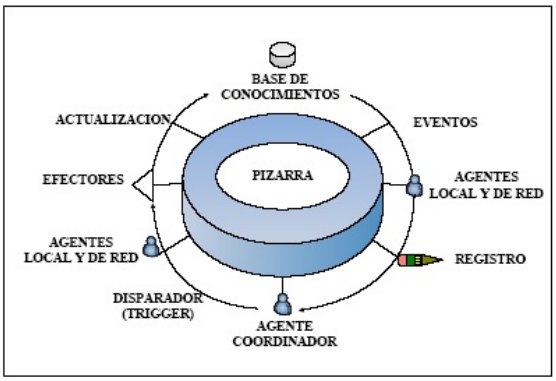


Figura 9. Ciclo de Control de Pizarra.

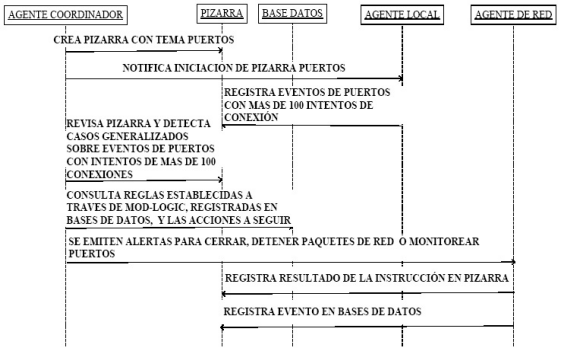


Figura 10. Diagrama de actividad para monitoreo de puertos.

comunicación en el sistema de agentes en base al ciclo de control que se presenta en la Figura 9.

### Ejemplo de uso del módulo pizarra

En el diagrama de actividad de agentes que se muestra en la Figura 10, se muestra como ejemplo, el intercambio de mensajes para monitoreo de la pizarra con el tema *puertos*, con el objetivo de detectar puertos abiertos que registren más de 100 intentos de conexión por un mismo ordenador en un período de 10 minutos.

### 2.3.3. Módulo Deliberativo-Cognitivo

Un aspecto importante en seguridad computacional que se toma en cuenta al momento de diseñar sistemas o herramientas de protección o de prevención, consiste en incluir componentes que permitan monitorear eventos, registrar en contenedores la información que se considera relevante en base a criterios previamente establecidos, aplicar técnicas de análisis para clasificar o reconocer patrones que permitan identificar comportamientos que puedan representar una amenaza, y aplicar procedimientos correctivos o preventivos cuando se requiere. Esta metodología, aunque exitosa, no ha logrado controlar la problemática "Zero day", y por lo tanto se asume que es necesario buscar alternativas que permitan automatizar y hacer más flexibles los procedimientos actuales, de tal forma que puedan implementarse y modificarse estrategias en tiempo de ejecución para atacar problemas de este tipo.

Tomando en cuenta lo antes expuesto y que los sistemas basados en agentes, entre otras cosas, pueden utilizar formas de razonamiento que les permitan interactuar y asumir comportamientos que definan el nivel y alcance de sus funciones, en la arquitectura ARSEC-AMS se incluye el módulo deliberativo-cognitivo, a través del cual pueden implementarse técnicas de reconocimiento de patrones, sistemas expertos, reglas basadas en lógica, etc.

### Ejemplo de uso del módulo deliberativo-cognitivo

Siendo la programación lógica una de las formas más eficientes y parecidas a las que el ser humano utiliza para representar el conocimiento, a continuación se presenta un ejemplo de un sistema de reglas que puede ser aplicado en este módulo donde se visualiza un posible escenario relacionado con el tema *puertos* y los siguientes supuestos semánticos:

**Supuesto 1** Se ha detectado que existe un *gusano*, que para efectos de ejemplificar, ha sido denominado *scanworm*, que se compone de la parte *cliente* y

la parte *servidora*, siendo ésta última la que se descarga en ordenadores para infectarlos y seguir propagándose a otros equipos que se encuentran cercanos al punto donde se encuentran localizados.

**Supuesto 2** La parte *cliente* es manejada por el autor del *gusano*, y se dedica a escanear la red buscando equipos que tengan abierto el puerto 2127, el cual es abierto en *listening* por la parte *servidora* que una vez activa se encuentra en espera de que la parte *cliente* se conecte.

**Supuesto 3** El objetivo de este *gusano* es establecer conexión entre ambas partes (*cliente* y *servidor*) con el fin de obtener información confidencial que está guardada en el equipo infectado.

**Supuesto 4** Se detectan anomalías en los equipos infectados provocados por este *gusano*, pero aún no se cuenta con la herramienta capaz de detectarlo, detenerlo y eliminarlo del sistema.

**Supuesto 5** Se ha propagado una alerta administrativa que permite registrar una regla en el módulo lógico, con el fin de cerrar el nodo de conexión, que es el puerto, y evitar la pérdida o robo de información hasta que se solucione el problema.

**Supuesto 6** Se registra la regla *scanworm* y se obtiene la siguiente plantilla:

**semántica:**

scanworm(n_puerto, t_operador, n_conexion, t_accion).
num_puerto(n_puerto).
num_conexion(n_conexion).
tipo_accion(t_accion).

**reglas:**

scanworm(n_puerto, t_operador, n_conexion, t_accion):
num_puerto(n_puerto),
num_conexion(n_conexion), tipo_accion(t_accion).

**datos:**

num_puerto('2127').
num_conexion('0').
tipo_accion('cerrar').
tipo_accion('revisar').

***Interpretación de la regla 'scanworm'***: Si el número del puerto es '2127' y el número de conexiones es '0', entonces se trata de 'scanworm' y se debe tomar la acción de 'cerrar' el puerto y 'revisar' el estado del mismo en períodos de tiempo predeterminados.

#### 2.3.4. Módulo de Seguridad y Reactivo

**Módulo de seguridad**. El módulo de seguridad trabaja conjuntamente con el módulo deliberativo-cognitivo, ya que permite establecer criterios y niveles de seguridad que pueden aplicar los agentes en determinados casos, e incluir datos relacionados con alertas, instrucciones y acciones que sirvan como elementos complementarios en la toma de decisiones.

**Módulo reactivo**. En el caso del módulo reactivo pueden programarse acciones que respondan a cierto tipo de eventos previamente fijados por el administrador o diseñador del sistema, siendo los agentes AL y AR quienes pueden asumir este rol. Como ejemplo de un agente reactivo, se puede tomar el caso de un agente que atiende la actividad de monitoreo de paquetes de red aplicando reglas derivadas de un árbol de decisión para descartar paquetes de red que puedan ser dañinos, o aplicando técnicas de reconocimiento de patrones para identificar texto que se incluye en las cabeceras de paquete de red que es considerado como un patrón que permite identificar a cierto tipo de *malware*.

## 3. Elementos básicos complementarios a la arquitectura ARSEC-AMS

Para efecto de complementar la arquitectura ARSEC-AMS, como elemento de comunicación, se ha diseñado la gramática *Content Language Security System Agent for Windows* (CLASS-W) [14], la cual ha sido diseñada para generar lenguajes de contenido y ser utilizada en el desarrollo de sistemas de seguridad basados en agentes, de acuerdo a las especificaciones fijadas por FIPA [12].

Por otra parte, se ha desarrollado la librería denominada Mod-Logic, Mod-Logic, traductor de predicados tipo Prolog a una base de datos relacional e intérprete de consultas con plantillas diseñadas con SQL [15], con la cual se puede guardar y recuperar predicados tipo Prolog en base de datos, con la misma funcionalidad que lo haría un compilador de prolog pero con la ventaja de que puede utilizarse en forma independiente en cualquier tipo de aplicaciones que permitan utilizar librerías de enlace dinámico.

Existen otro tipo de elementos diseñados para ser implementados en la arquitectura ARSEC-AMS, como por ejemplo, el procedimiento de cifrado de información a través de intercambio de clave pública y privada, pero que por cuestiones de espacio, al igual que ejemplificar todos los módulos no pueden ser descritos en este documento.

Por último, cabe mencionar que el contenido de este

artículo forma parte del trabajo que se realiza dentro de un proyecto de investigación en materia de seguridad computacional. Con la implementación de la arquitectura ARSEC-AMS se han desarrollado herramientas que han sido motivo de cuatro tesis de licenciatura.

## 4. Conclusiones

La seguridad en las redes de computadoras es un tema que cada día cobra más importancia debido a la gestión masiva de datos y a la generalización del uso de tecnología, lo cual ha permitido a todo tipo de organismos aumentar su productividad y riqueza pero también se ven obligados a buscar medios de protección que eviten pérdidas de información que generalmente tienen repercusiones financieras significativas.

A pesar del avance que existe en alternativas de protección en materia de seguridad computacional, como lo es el desarrollo de herramientas software como antivirus, sistemas detectores de intrusos, etc., la problemática denominada "amenazas de día cero" o "zero day" es la que más preocupa a los expertos en seguridad, ya que la naturaleza de este problema consiste precisamente en el surgimiento constante de nuevas amenazas que el software de protección no puede detectar porque no han sido diseñadas para inferir eficientemente nuevas amenazas y desconocen patrones de comportamiento ajenos a los ya registrados.

Con un enfoque distinto a los actuales, la arquitectura ARSEC-AMS, como herramienta de diseño, permite a los expertos, a un administrador o desarrollador de sistemas de seguridad, administrar o diseñar y desarrollar sistemas basados en tecnología de agentes aprovechando la experiencia adquirida en materia de seguridad.

Por último, hay que agregar que si bien existen diferentes tipos de gramáticas para desarrollo de sistemas de agentes, no se encontró evidencia de alguna gramática que en forma rápida y eficiente permita el diseño y desarrollo de sistemas de agentes para resolver problemas complejos en materia de seguridad computacional. Por ello, se concluye que los elementos creados para implementarse en la arquitectura ARSEC-AMS, como lo es la gramática CLASS W y la librería Mod-Logic son componentes básicos, pero no necesariamente son elementos obligatorios para el uso de la arquitectura, ya que pueden utilizarse gramáticas de agentes existentes y métodos que permitan controlar el sistema de razonamiento de los sistemas basados en tecnología de agentes.

## Referencias


[1] J.M. Bradshaw. (1997), *Software Agents*, http://agents.umbc.edu/introduction/01-Bradshaw.pdf.

[2] La Flecha. (Enero, 2006), *Diario de Ciencia y Tecnología, El coste de los virus informáticos asciende a 92,000 millones de euros en 10 años (110,000 millones de dólares)*, Agencia EFE, http://laflecha.net/canales/seguridad/200601301.

[3] McAfee Proven Security. (Diciembre, 2006), *Microsoft Word 0-Day Vulnerability III*, http://vil.nai.com/vil/Content/v_vul27264.htm.

[4] R. D. Pethia. (Septiembre, 2003), "Viruses and Worms: What Can We Do About Them?," Software Engineering Institute, Carnegie Mellon University, CERT®Coordination Center, Pittsburgh, Pennsylvania, USA, http://archive.cert.uni-stuttgart.de/isn/2003/09/msg00092.html.

[5] El Universal.com.mx. (Junio, 2007), "Se propaga Gusano en YouTube," http://www.el-universal.com.mx/articulos/40642.html.

[6] R. Clarke. (Julio, 2004), "Cómo afrontar los cambios radicales de la seguridad en el ciberespacio," http://www.symantec.com/region/mx/enterprisesecurity/content/expert/LAM_4198.html.

[7] B. Schneier. (2004), "SIMS: Solution, or Part of the Problem?," *IEEE Security and Privacy*, Vol. 2, No. 5, pp. 88, ISSN: 1540-7993, http://doi.ieeecomputersociety.org/10.1109/MSP.2004.83.

[8] D. Kinny, M. Georgeff and A. Rao. (1997), "A Methodology and Modelling Technique for Systems of BDI Agents," Report, http://www.dia.fi.upm.es/~phernan/AgentesInteligentes/referencias/kinny96methodology.pdf.

[9] F. Zambonelli, N. Jennings and M. Wooldridge, "Developing multiagent systems: the Gaia Methodology," *ACM Trans on Software Engineering and Methodology*, Vol. 12, No. 3, pp. 317-370, 2003.

[10] A. Garro, "GAIA.Modeling Notation Source, Version 03-03-1217:25EST," http://www.auml.org/auml/documents/Gaia.doc.

[11] F. Ingrand and M. Georgeff. (1990), *Managing Deliberation and Reasoning in Real-Time AI Systems*, In Proceedings of the Workshop Innovative Approaches to Planning, Scheduling and Control, pp. 284-291, http://books.google.com.mx.

[12] FIPA, "Foundation for Intelligent Physical Agents," http://www.fipa.org.

[13] N. Carver and V. Lesser, "The evolution of blackboard control architectures," Technical report, University of Massachusetts Amherst, pp. 19-23, 1992, http://www.cs.siu.edu/~carver/ps-files/tr92-71.ps.gz.

[14] G. Cota, P. Flores and J. Suárez, "CLASS-W, a Grammar for Security System Development based on Environment Equipment Agents with Windows-Technology NT," *Research in Computing Sciences. Advances in Computer Science and Engineering*, ISSN: 1870-4069, pp. 65-76, 2009, http://prometeo.cic.ipn.mx/2009/index.html.

[15] G. Cota, P. Flores, I. López and M. Morales, "Mod-Logic: Un Traductor de Predicados Tipo Prolog a una Base de Datos Relacional e Intérprete de Consultas con Plantillas Diseñadas con SQL," *Revista Iberoamericana de Sistemas, Cibernética e Informática*, ISSN: 1690-8627, Vol. 5, No. 2, pp. 25-29, 2008, http://www.iiis.org/iiis/best_paper.asp?year=2008.