Entre Ciencia e Ingeniería, ISSN 1909-8367
Año 10 No. 20 - Segundo Semestre de 2016, página 24 - 31

Sistema multi-agente deliberativo para la obtención y análisis de datos en Honeynets[1]

Deliberative multi-agent system for data collection and analysis in Honeynets

Sistema multi- agente deliberativo para a obtenção e análise de dados em Honeynets.

E. M. Giraldo y G. A. Isaza

Recibido Noviembre 10 de 2015 – Aceptado Mayo 30 de 2016

***Resumen*— Este artículo presenta la propuesta e implementación de un sistema multi-agente deliberativo para la extracción de los diferentes eventos capturados dentro de una HoneyNet, su análisis, procesamiento y uso dentro de un modelo de razonamiento basado en casos. Se presenta el diseño general de la plataforma multi-Agente, la responsabilidad de cada uno de sus componentes, su ejecución y los resultados obtenidos**

***Palabras clave*— Honeynet, IDS, agentes deliberativos, sistema multiagente.**

***Abstract*— This paper aims to present the proposal and implementation of a deliberative multi-agent for the extraction of the different events captured within a Honeynet, its analysis, processing and use within a case based reasoning model. The overall design of the Multi-Agent platform, the responsibility of each of its components, its implementation and the results obtained are presented.**

***Key words*—Honeynet, IDS, deliberative agents, multiagent system.**

***Resumo* - Este artigo apresenta a proposta e implementação de um sistema multi-agente deliberativo para a extração dos diferentes eventos capturados dentro de uma Honey Net, sua análise, processamento e uso dentro de um modelo de raciocínio baseado em casos. Apresenta-se o desenho geral da plataforma multi- agente, a responsabilidade de cada um dos seus componentes, sua execução e os resultados obtidos.**

***Palavras chave* – Honeynet, IDS, agentes deliberativos, sistema multi-agente.**

I. Introducción

Las plataformas de prevención de intrusiones basan su funcionamiento en la comparación del evento registrado contra un sistema de información, plataforma o cualquier componente que esté conectado a una red de datos, con firmas previamente identificadas; de esta manera pueden llegar a determinar si una acción que se realiza contra un sistema puede categorizarse como un posible evento malicioso. El entorno en el cual se identifican los eventos maliciosos y se generan las firmas correspondientes, es normalmente un ecosistema computacional vulnerable, donde, mediante la utilización de una arquitectura de referencia como es el proyecto *HoneyNet*[2], implementan un tipo específico de *HoneyPot*[3], exponiendo, en internet, los entornos vulnerables para identificar los diferentes comportamientos de atacantes, los componentes de software que lanzan, los pasos que ejecutan y cómo logran acceder o extraer información de este sistema. El núcleo de esta propuesta se centra en la implementación de una *HoneyNet* basada en agentes

[1]Producto derivado del proyecto de investigación "Sistema multi-agente deliberativo para la obtención y análisis de datos en Honeynets". Presentado por estudiante de Maestría en Gestión y Desarrollo de Proyectos de Software de la Universidad Autónoma de Manizales.

E. M. Giraldo, estudiante, de la Universidad Autónoma de Manizales, Manizales (Colombia); email: erikumanizales@gmail.com.

G. A. Isaza, Profesor Asociado Grupo GITIR - Ci²Dt² Facultad de Ingeniería de la Universidad de Caldas, Manizales (Colombia); email: gustavo.isaza@ucaldas.edu.co.

[2]Es una arquitectura cuyo objetivo es plantear los componentes necesarios para exponer de manera consciente a atacantes, sistemas, aplicaciones y servicios reales, el acceso ilícito a estos, para la captura de comportamientos y formas de ataque.

[3]Componente de software, cuyo objetivo es atraer atacantes, mediante la simulación de elementos vulnerables, lo anterior tiene como propósitos la recolección de información e investigación.

inteligentes con capacidad de deliberación para aproximar la identificación de nuevos patrones.

El artículo se compone de la definición del problema, la justificación del proceso y los requerimientos, la metodología utilizada para el desarrollo del proyecto y la implementación del modelo con los resultados obtenidos.

II. Estado de la cuestión

En la actualidad las empresas tienen desplegadas sus plataformas en redes públicas, de manera transparente o a través de capas medias de seguridad. Para que una entidad pueda llegar a identificar de manera eficiente -a través de un software o plataforma- eventos maliciosos depende en gran medida de que la fuente de conocimiento que lo alimente esté actualizada, en la mayoría de los casos, funcionan a partir de firmas que describen e identifican eventos maliciosos. La captura de estas firmas puede llevarse a cabo dentro de una red HoneyNet, un tipo específico de HoneyPot de alta interacción, que expone máquinas vulnerables y componentes específicos que permiten la captura y análisis del comportamiento de atacantes. El proyecto HoneyNet define una arquitectura o diseño de referencia para su implementación, dicha arquitectura inicial se define de la siguiente forma en la figura 1:

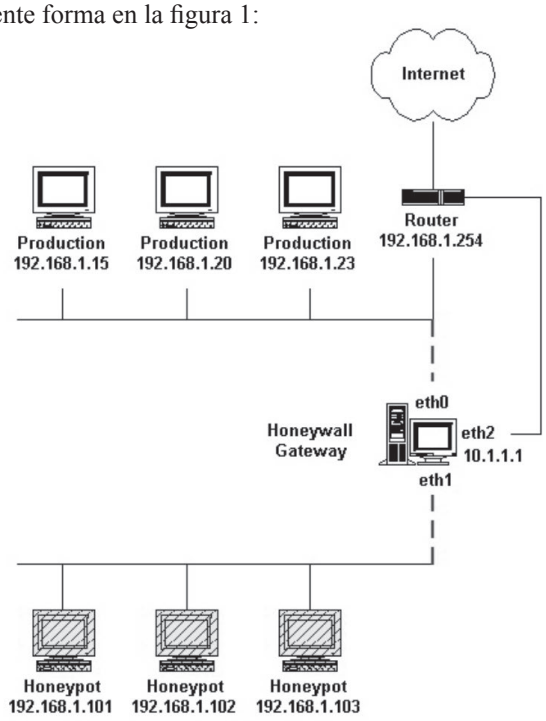

Fig. 1. Arquitectura base de una HoneyNet [1]

El diseño anterior está concebido para ser un sistema centralizado con un único punto de adquisición de datos, esto significa que:

- El límite en la captura de nuevos patrones estaría en un solo elemento, implicando un porcentaje realmente bajo de captura de ataques conocidos y desconocidos.
- Habría una única arquitectura implementada y por consiguiente un único objetivo estructurado.
- Se tendría un único objetivo dentro de millones de máquinas reales y vulnerables.
- El acceso a los datos recolectados sería limitado.

Debido a lo anterior, el uso de un sistema multi-agente como base fundamental del sistema distribuido permite que estos, a diferencias de arquitecturas distribuidas tradicionales, tengan un comportamiento basado en su entorno, establecidos en una semántica de comunicación y en sus características principales, donde todos trabajan para cumplir un objetivo, en este caso, extraer información de diferentes fuentes usando la misma semántica y deliberar sobre los datos obtenidos para la posible identificación de atacantes.

El boletín de seguridad de Kaspersky *2014"* muestra que los ataques tienden a incrementarse. Basándose en la comparación con el año previo, algunas estadísticas son:

- 6,167,233,068 ataques fueron detectados y neutralizados.
- 38% de usuarios fueron objetivo de un ataque web al menos 1 vez en el año.
- 1,910,520 intentos de lanzamiento de aplicaciones maliciosas de bancos en las máquinas de usuarios fueron neutralizados.
- 1,432,660,467 ataques fueron lanzados desde recursos en línea [2].

Lo anterior ha permitido que surjan iniciativas como el proyecto HoneyNet, que además de definir los lineamientos para implementar, de la mejor forma una arquitectura para la investigación de ataques y obtención de firmas, trata de contribuir en la identificación de nuevos eventos o elementos maliciosos para que puedan ser prevenidos mediante el uso de Sistema de Detección de Intrusiones (IDS). Este tipo de iniciativas, ha planteado soluciones que permiten la captura de eventos dentro de una única red vulnerable y controlada, además, han sido concebidas de manera centralizada, limitando en gran medida la cantidad de eventos capturados, siendo estas de por sí, excelentes alternativas.

Debido a que los **IDS** están pensados para estar centralizados en una sola estación, lo cual es una gran limitante de los mismos, se han propuesto soluciones basadas en sistemas distribuidos. Los principales problemas de utilizar arquitecturas tradicionales son [3]:

- La consola central tiende a ser un solo punto de fallo, en tal sentido, la red entera está sin protección si ésta falla.
- La escalabilidad es limitada. Procesar toda la información en un nodo implica un límite en el tamaño de la red que puede monitorizar.
- Se evidencian dificultades en reconfigurar o añadir capacidades y firmas a los **IDS.**
- El análisis de los datos de la red puede ser defectuoso, debido a su arquitectura centralizada, durante un periodo de fallo, los datos capturados pueden llegar a ser defectuosos o presentar pérdida de paquetes.

Así pues, la propuesta del uso de un sistema multi-agente como base fundamental del sistema distribuido, permite que estos, a diferencias de arquitecturas distribuidas tradicionales, tengan un comportamiento basado en su entorno, establecidos en una semántica de comunicación y en sus características principales, donde, todos trabajan para

cumplir un objetivo, en este caso, extraer información de diferentes fuentes usando la misma semántica y deliberar sobre los datos obtenidos para la posible identificación de atacantes.

A continuación se presentan algunos de los trabajos más significativos que se han realizado en el desarrollo de sistemas multi-agentes y la seguridad informática.

Una de las áreas en las que se ha venido trabajando es en la visualización de la información de grandes volúmenes de datos de HoneyNets dentro de una red comprometida, la cual es visualizada usualmente por elementos de monitoreo mostrando patrones en el trafico [4]. También se ha trabajo en modelos que permiten dar proyección para visualizar los datos de tráfico [5]. Estudios en los que se demuestra la importancia de las HoneyNet en la seguridad informática, una vez analizadas las definiciones más comunes, la planificación, implementación y casos de prueba para la implementación de una HoneyNet se pueden evidenciar en [6]; al igual, cómo la recopilación de datos de éstas y la aplicación de técnicas de minería de datos han dado como resultado nuevas reglas asociadas a datos históricos o estructuras específicas a ser utilizadas por **IDS** o *Firewalls* [7]. Los lineamientos y guías del proyecto HoneyNet han sido definitivos en la concepción de una arquitectura centralizada y suficientemente robusta, para realizar la implementación de HoneyNets, que sea un elemento fundamental dentro del macro de la seguridad informática [8]. El uso de sistemas multi-agente para la reducción de falsas alarmas dentro de sistemas de detección de intrusos [9] o el aumento de la atención a atacantes para que accedan a la red HoneyNet [10] son puntos que se han trabajado, sin embargo, tienen focos diferentes en cuanto a que ninguno centra sus esfuerzos en la detección de nuevos patrones o extracción distribuida de datos. También se ha trabajado en el uso de un sistema multi-agente, con la inclusión de un sistema de razonamiento basado en casos asentado en creencias, deseos e intenciones para la detección y clasificación de ataques a partir de la técnica de detección anómala [11] o en la utilización de un sistema multi-agente que se encarga de remover procesos maliciosos y archivos ejecutables de servidores, protegiendo la red de posibles ataques *0-day* [12].

Iniciativas más recientes como [13] y [14] evidencian una taxonomía de HoneyNets basada en sistemas de razonamiento adaptativo que permiten clasificar y deliberar sobre algunos casos de redes señuelo. En [15] se presenta un sistema basado en deliberación usando *JColibri*, no obstante, sobre una arquitectura de un solo nodo y no como un entorno distribuido que conforme una HoneyNet. En [16] y [17] se proponen clasificaciones de eventos de ataques basados en sensores HoneyNets y en [18] un análisis de redes señuelo basada en técnicas de reconocimiento de patrones.

En términos generales los proyectos e investigaciones homogéneos concentran sus esfuerzos en la identificación de ataques en diferentes momentos, componentes y capas, obteniendo o capturando paquetes a nivel de red para su procesamiento, pero no han buscado nuevos patrones de ataque dentro de los ya existentes, teniendo en cuenta que un evento por sí mismo no necesariamente identifica un vector de ataque.

III. Materiales y métodos

El proceso que se siguió para realizar la implementación de todo el proyecto se apoyó en tres aristas: en la primera se conceptualizó como el problema, se identificó la solución y se expresaron un conjunto de requerimientos; la segunda trata de garantizar el correcto modelamiento de los componentes de software y hardware necesarios para obtener un entorno completo y cumplir con la solución; la tercera surge para garantizar el desarrollo organizado de cada uno de los programas o componentes de software necesarios.

Para alcanzar el modelamiento correcto, se utilizó **UPSAM**[4] como metodología de Ingeniería de Software de Agentes, que enmarca el proceso de análisis de sistemas multi-agentes en el paradigma del modelado de agentes y permite alinearlo al proceso de diseño de **RUP**[5].

Se ha hecho uso de esta metodología para la realización de un análisis y diseño organizado de un sistema orientado a agentes, contextualizándola en el uso de algunos procesos de **RUP** (Proceso Unificado de *Rational*), orientados al desarrollo de agentes, dirigido por casos de uso y centralizado en la arquitectura. A continuación se muestran algunos modelos asociados a **UPSAM**:

- Modelo de tareas: servicios, tareas, objetivos y funcionalidades asociados a roles. Este modelo normalmente se expresa a través del uso de casos de uso orientado a agentes.
- Modelo de arquitectura de la organización de agentes: descripción de agrupación de los agentes y sus relaciones. En este modelo es normal la utilización de un diagrama general de actividades acompañado de la especificación de responsabilidades, propósitos, percepciones, tareas y salidas.
- Modelo de agentes: descripción de agentes a instancias, comportamiento y cómo realizar el control especificado, en este modelo es común la utilización de diagramas de clases basados en los conceptos claves de los roles asociados a los agentes.
- Modelo de comunicación de agentes: descripción de la secuencia de interacción entre los diferentes tipos o roles, este modelo es normalmente especificado a través de diagramas de secuencia [19].

Teniendo un modelamiento correcto que apoya directamente el diseño de la plataforma, se adaptó **PSP**[6]**0,** que permite de manera individual un seguimiento al proceso de desarrollo, determinando entregables básicos, métricas y elementos de seguimiento. Se han explicado las diferentes fases del proceso PSP en su versión cero, al igual que los artefactos asociados a cada etapa [20]. Para el desarrollo de este proyecto se optó por la división del proyecto en

[4]Lenguaje y Metodología de Modelamiento UML, focalizado a agentes inteligentes, Castillo A.G et al 2004. Agentes para la Gestión de Conocimiento.

[5]Proceso de desarrollo de software que proporciona un conjunto de pasos y metodologías aplicables/ adaptables a las necesidades de cada organización y/o proyecto.

[6]Proceso para la construcción de software, formado por métodos, formularios y scripts que le presentan a los desarrolladores como planear, medir y gestionar su trabajo.

micro-desarrollos, acorde a lo expuesto para proyectos más extensos [21].

A. Especificación de requerimientos

El esquema de especificación, se compone de:

- Rol: define qué rol bajo el sistema es el interesado en el cumplimiento o logro de una necesidad especifica.
- Necesidad: la definición de qué se espera lograr con la implementación de un punto específico.
- Objetivo: lo que se desea lograr u obtener posterior a la implementación.
- Criterio de la aceptación: los elementos que se esperan obtener una vez se realiza la implementación y que garantizan que se está logrando el objetivo de la misma.

Algunos ejemplos de cómo se realizó el plasmado de estos son:

Tabla I

Especificación de requerimientos del proyecto

Rol	Necesidad	Objetivo	Criterios de Aceptación
Usuario	Obtener los posibles ataques que se detectan en la red	Generar y alimentar la base de casos	1. Consulta SQL eficiente 2. SQL preciso al esquema utilizado por SNORT[7] 3. SQL que permita realizar la consulta asociada
Usuario	Identificar de cada ataque la IP origen, la IP destino, el evento asociado y la ocurrencia	Generar y alimentar la base de casos	1. Consulta SQL eficiente 2. SQL preciso al esquema utilizado por SNORT 3. SQL que permita realizar la consulta asociada
Usuario	Tener una aplicación liviana, desacoplada de la arquitectura del SO que permita a consultar los ataques y obtener los campos especificados	Garantizar de manera masiva su implementación para la obtención de vectores de ataque geográficamente distribuidos	1. Aplicación Java 2. Ejecución de la aplicación con una máquina virtual estándar en un sistema operativo Windows 3. Ejecución de la aplicación y visualizar la consulta de datos 4. Ejecución de la aplicación y búsqueda de servicio de construcción de casos

B. Diseño

Las características asociadas de autonomía, reactividad, proactividad, movilidad, inteligencia, entre otras, sugieren y sustentan la necesidad de dividir, de manera sistémica, los diferentes actores que interactúan con las diferentes capacidades asociadas a los agentes. Lo anterior soporta la interacción compleja entre los diferentes agentes para lograr un objetivo común. El diseño asociado de los agentes que intervienen en la plataforma y sus responsabilidades, se presenta en la Tabla II:

[7] Sistema de detección de intrusiones.

Tabla II

Responsabilidades de los Agentes

Agente	Responsabilidades
Poller	Se encarga de sensar y obtener paquetes capturados previamente por el ambiente dentro de la *HoneyWall*. Este agente es clonado y distribuido dentro de las diferentes *HoneyWalls*, registrándose en cada caso en las *YellowPages*.
Builder	Se encarga de sensar y recibir la información referente a posibles ataques o comportamientos anormales identificados previamente por agentes *Pollers*, además, notifica al agente deliberador sobre nuevos datos. Este agente es clonado y distribuido de manera estratégica de acuerdo a la carga y posición geográfica asociada
Maa-UI	Se encarga de sensar y recibir notificaciones sobre nuevos ataques procesados por el agente *Builder*. También se encarga de controlar la interacción entre el **UI** y la lógica propia del agente, deliberar y persistir nuevos casos. El agente encargado de la deliberación es lanzado desde el UI de este.

La arquitectura general planteada para la implementación de la plataforma MAS+CBR se definió en 4 niveles, con opción de escalabilidad horizontal de manera transparente, como se presenta en la Figura 2, el comportamiento de los agentes asociados, se describe como:

- Agente Maa-Ui o Sistema de razonamiento y gestor de conocimiento.
- Interfaz de usuario que permite la gestión y configuración del CBR.
- Elementos de negocio que garantizan la interacción hacia los demás componentes.
- Elementos de integración con otras plataformas y componentes.
- Elementos de persistencia que trabajan hacia la base de datos.

Agente Builder o constructor y procesador de datos.

- Elementos propios de la definición del agente.
- Elementos de negocio que garantizan la interacción con otros componentes.
- Elementos y servicios comunes a los agentes.

Agente Poller o extractor de datos.

- Elementos propios de la definición del agente.
- Elementos de negocio que garantizan la interacción con otros componentes.
- Elementos y servicios comunes a los agentes.

El comportamiento deliberativo asociado al agente Maa-Ui se diseñó basado en un CBR (Case Base Reasoning) así:

a. RETRIEVE u obtención, esta fase se encarga de obtener casos previos de la base de conocimiento, acorde a la configuración base realizada. En este punto la fuente de datos ha sido el esquema propio del IDS, el cual fue consultado por el agente extractor y entregado para su procesamiento a un agente constructor.

b. REUSE o reutilización, con base a la información obtenida se revisa y plantea una propuesta de solución al problema dado. Dentro de la obtención de los resultados se realizó una configuración base de parámetros de peso y similitud para su comparación con la base de conocimiento obtenida del IDS, esto da como resultado un posible escenario que resuelve el problema planteado dentro de la búsqueda de nuevos comportamientos o vectores de ataque.
c. REVISE o revisión, se realiza una aprobación del planteamiento de solución. En este punto los casos obtenidos y acordes al planteamiento de la hipótesis del proyecto, son aprobados de manera automática, no siendo esto una condición de fondo, siendo posible crecer en la revisión.
d. RETAIN o retención, se almacenan los casos que se consideran útiles.

Fig. 2. Arquitectura de la Plataforma

En la fase de *Reuse, el agente* deliberativo se encarga de clasificar los nuevos escenarios con base en los previamente obtenidos de la base de conocimiento, para lo cual, utiliza el algoritmo KNN:

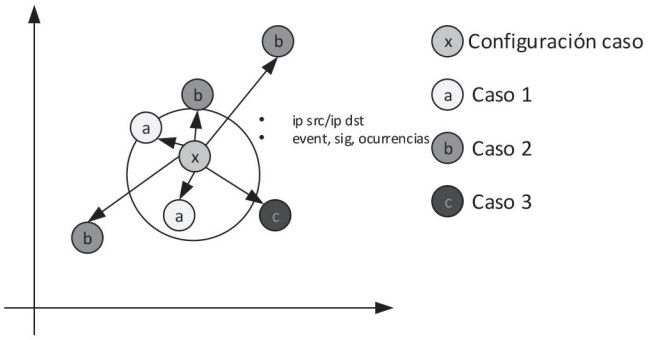

Fig. 3. Algoritmo KNN de Clasificación.

Del comportamiento que se presenta en la gráfica se pude concluir que la configuración *x* definida previamente, se encuentra acorde a la proximidad cerca de los elementos *a, b* y *c*, siendo éstas, posibles tipificaciones para la misma, sin embargo y debido a la cantidad de elementos cercanos se clasifica como (**a**). La configuración que se usa para la propuesta se basa en los siguientes elementos, definidos previamente en el modelo de dominio:

- IP Origen
- Ip Destino
- Alertas
  - Identificación de Alertas
  - Ocurrencias
  - Fecha

Durante el proceso de ejecución del algoritmo KNN, las funciones de similitud son críticas en la comparación, en este caso, se realizó la implementación de una función específica de comparación, así:

$$Sea\ A = \{Alerta\ 1, Alerta\ 2, \dots., Alerta\ n\} \quad (1)$$

$$Sea\ B = \{Alerta\ 1, Alerta\ 2, \dots Alerta\ n\} \quad (2)$$

Donde A son las alertas asociadas a patrones de ataques ya identificados y B un nuevo patrón configurado:

$$A \cap B = \{Alerta\ 1, Alerta\ 3\} \quad (3)$$

La intersección de los conjuntos define los elementos comunes a ambos, por consiguiente, si *n* es el total de elementos del conjunto *A* y *m* es el total de elementos del conjunto resultante de la intersección

$$A \cap B = \{Alerta\ 1, Alerta\ 3\} \quad (3)$$

Siendo SV el resultado de similitud y realizando una aproximación sobre el valor del resultado.

C. Implementación

El desarrollo de la plataforma sesga su operación en la extracción de datos de una *HoneyNet*, esta se encuentra implementada bajo un esquema de virtualización básico que involucra los siguientes elementos:

1) *Máquina Atacante: esta máquina virtualizada con el único objetivo de servir como host atacante, desde el cual se realizarían ataques a máquinas vulnerables dentro del laboratorio.*
2) *HoneyWall: esta máquina fue virtualizada para que actuará como un punto de captura centralizada dentro de la red de posibles ataques, su objetivo es permitir el control de los datos de entrada que pueden indicar un ataque, capturar estos datos, analizarlos y permitir herramientas para analizarlos.*
3) *Máquina(s) Atacante(s): VM (Virtual Machines) para representar una o varias máquinas vulnerables, desde su sistema operativo hasta las aplicaciones que corren.*

Con la implementación a nivel de hardware virtualizado, se desarrolló el sistema multi-agente, modelado en la figura 4.

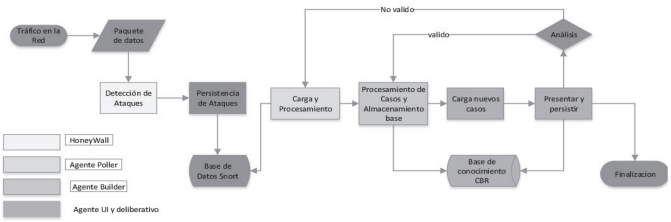

Fig. 4. Diagrama Estado Básico CBR + MAS (Tareas Generales del Sistema).

Los eventos relevantes que son identificados y transcritos a representaciones JSON para ser enviados al agente disponible, cuentan con la siguiente estructura:


```
{"sid":1,"cid":292,"sigName":"ET CURRENT_EVENTS Exploit Kit Delivering
Compressed Flash Content to Client","signatureId":512,"timestamp":"ago 25,
2013","sigPriority":2,"sigRev":1,"eventsCount":21,"ipSrc":"201.232.123.9","ipDs
t":"192.168.0.13","ipVer":4,"ipHlen":5,"ipTos":0,"ipLen":1500,"ipId":22051,"ipF
lags":0,"ipOff":0,"ipTtl":56,"ipProto":6,"ipCsum":57681
```


El Agente Poller es el encargado de realizar la consulta y el envío de los datos a los agentes builder que se encuentran disponibles:

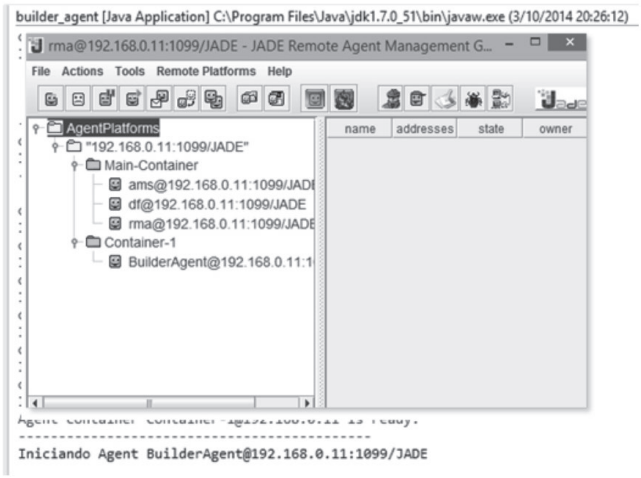

Fig. 5. Agente Builder Jade RMA.

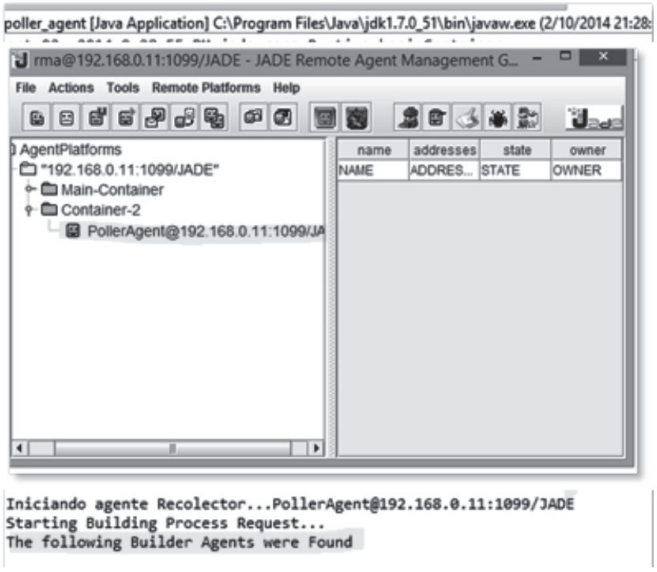

Fig. 6. Ejecución Agente Poller RMA Jade.

Una vez el agente Builder recibe la información, la procesa y almacena en la estructura específica de la plataforma. Para que el agente deliberativo inicie su proceso, el agente builder le notifica el Mensaje ACL. La interacción base se puede resumir en la siguiente figura, capturada a través del agente sniffer dentro framewok JADE:

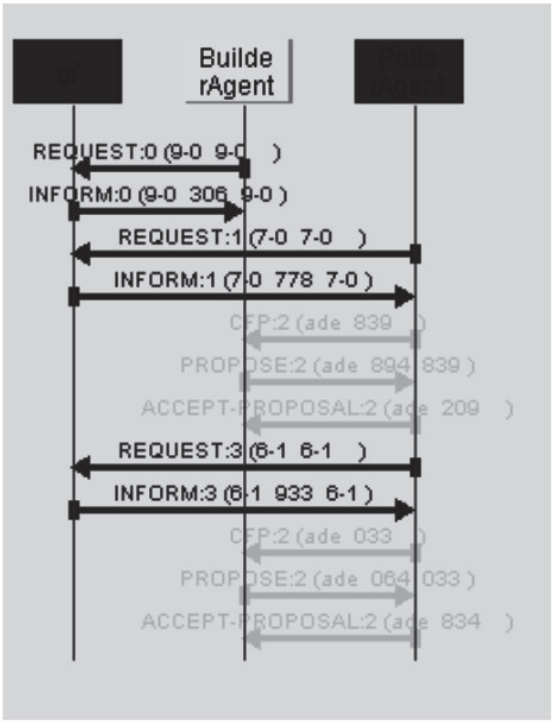

Fig. 7. Agente Sniffer.

- Tanto el agente Builder como el Agente Poller se registran en las páginas amarillas ofreciendo servicios específicos.
- Entre los agentes se inicia una interacción, donde el agente Poller solicita al agente Builder encontrado el procesamiento de la información, una vez acepta, ésta es recibida y procesada.

El agente Maa-ui, una vez configurado, está pendiente de la recepción de nuevos eventos, para esto, el agente builder una vez procesa un segmente de información, realiza la notificación necesaria.

III. Resultados

Con la ejecución del SMA (Ver Figura 8), se dio una aproximación al fortalecimiento de los métodos de extracción de información de diferentes entornos en el marco de la seguridad, basada en redes señuelo, siguiendo los lineamientos del proyecto HoneyNet y el núcleo de eventos fuera la misma, logrando una solución no intrusiva pero sí dependiente. La ejecución del prototipo, como se puede observar en la Figura 9, presenta una tasa de precisión -de detección- viable (Ver Tabla III); que puede sesgarse por las funciones de similitud; en este caso las asociadas a los eventos de la la configuración para detectar nuevos vectores de ataque se basa en el algoritmo KNN y el modelo CBR confiable (Ver tabla IV).

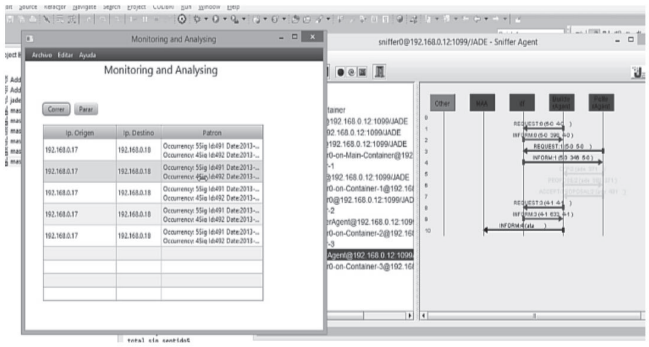

Fig. 8. Resultado Ejecución Sistema Multi-Agente.

Tabla III
Detección correcta vs detección fallida

	Config 1	Config 2	Config 3	Config 4
Vectores Acertados	90%	20%	90%	30%
Vectores Incorrectos	10%	80%	10%	70%

En la tabla IV se muestra el comportamiento de los patrones detectados.

Tabla IV
Configuración base cbr

Configuración	ip_src	dst_ip	Alerts	Función
Configuración 1	1	1	1	Cadena|Set
Configuración 2	1	1	0,5	Cadena|Set
Configuración 3	0	1	1	Cadena|Set
Configuración 4	1	0	0,5	Cadena|Set

En la figura 9, se presenta el resultado de detección correcta e incorrecta de nuevos casos

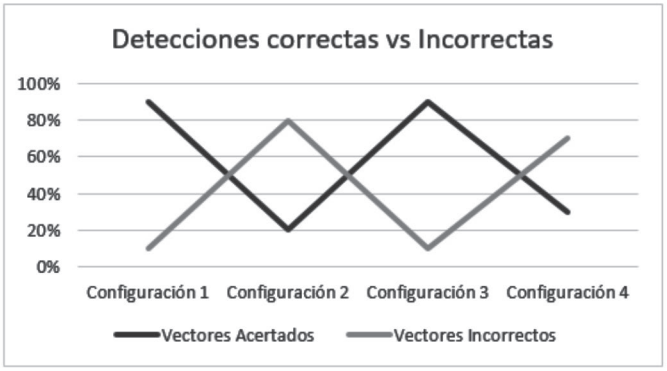

Fig. 9. Comportamiento de la detección.

El comportamiento anterior muestra una alta asertividad en las configuraciones de clasificación, donde los atributos son cadenas de texto fragmentadas; con un comportamiento diferente en el momento que se utiliza la función de configuración asociada al listado de alertas, donde se utilizó la intersección de los objetos sin especificar el detalle de la comparación.

IV. Conclusiones

Con la implementación del sistema multi-agente se lograron identificar diferentes tipos de vectores de ataques e información anómala, con una base de conocimiento previamente cargada a partir de eventos de diferentes entornos, con una deliberación acertada de nuevos patrones de intrusión. El sistema multi-agente deliberativo se basó en la implementación de un CBR encargado del razonamiento de los agentes, sin embargo, las funciones de comparación que se desarrollaron han tenido en algunos casos, comportamientos no acertados al momento de realizar la ejecución del algoritmo de clasificación, por lo cual, la optimización debería aumentar la capacidad de deliberación correcta tanto de vectores de ataque como de información anómala.

Si bien la implementación de un algoritmo de aprendizaje supervisado como es CBR con un algoritmo de clasificación KNN para obtener un resultado de deliberación sobre información recolectada presentó un comportamiento aceptable; dependía de una buena base de conocimiento, que para este caso, no estaba afinada en un porcentaje superior al 80% y reduciendo los falsos positivos, generando en algunos casos falsas deliberaciones sobre nuevos vectores de ataque y datos anómalos.

Referencias

[1] HoneyNet Project, “Know your enemy: HoneyNets. Honeynet Project” 2006. Disponible en: http://old.honeynet.org/papers/honeynet/
[2] M. Garnaeva, V. Chebyshev, D. Makrushin, R. Unuchek, A. Ivanov, “Kaspersky Security Bulletin 2014. Overall statistics for 2014” Securelist, 8, Dic.2014. Disponible: https://securelist.com/analysis/kaspersky-security-bulletin/68010/kaspersky-security-bulletin-2014-overall-statistics-for-2014/
[3] Isaza, G. Castillo, AG., et al. “Towards Ontology-Based Intelligent Model for Intrusion Detection and Prevention”. En: Estados Unidos Journal Of Information Assurance And Security ISSN: 1554-1010 ed: v.5 fasc.2 p.376 - 383 ,2010
[4] R.A. Becker, S.G. Eick, and A.R. Wilks, “Visualizing Network Data,”*IEEE Transaction.* Visualization and Computer Graphics, vol. 1, no.1, pp. 16-28, Mar. 1995.
[5]A. Alonso, S. Porras, E. Ezpeleta, E. Vergara, I. Arenaza, R. Uribeetxeberria, and E. Corchado, “Understanding Honeypot Data by an Unsupervised Neural Visualization,”*Springer Berlin Heidelberg*. Computational Intelligence in Security for Information System 2010, vol. 85, pp. 151-160,2010.
[6] C. Döring, “Improving network security with Honeypots, Honeypots Project”Tesis de Maestría dirigida por Dr. Heinz-Erich Erbs,Universidad Wisconsin- Platteville. Darmstadt. 2005.
[7] J. Yin, G. Zhang, andY-Q. Chen, “Intrusion discovery with data mining on Honeynet”, 2003 International Conference Machine Learning and Cybernetics”, pp.41-45,2-5 Nov. 2003.
[8] G. Rammidi, (2010). “Survey on Current Honeynet research”. Disponible: http://honeynetproject.ca/files/survey.pdf
[9] B. Khosravifar, M. Gomrokchi, and J. Bentahar, “A Multi-Agent-Based Approach to Improve Intrusion Detection Systems False Alarm Ratio by Using Honeypot,” International Conference on Advanced Information Networking and Application, pp.97-102, 2009.
[10] H. Wang, H andQ. Chen, Q, “Design of Cooperative Deployment in Distributed Honeynet System,”14th International Conference on Computer Supported Cooperative Word in Design (CSCWD), pp.711-716,2010.
[11] A. Herrero and E. Corchado, “Multiagent System for Network Intrusion Detection,” Computational Intelligence in Security for Information System, pp.143-154, 2009. Springer Berlin Heidelberg.
[12] I.S. Kim andM.H. Kim, “Agent-Bases Honeynet Framework for

Protecting Server in Campus Networks,"Information Security, IET, vol. 6, no. 3, pp-202-211, 2012.
[13] Fan, W., Du, Zhihui, Fernandez, D. Taxonomy of Honeynet Solutions. Conference: SAI Intelligent Systems Conference (IntelliSys), At London, United Kingdom. 2015
[14] Pauna, A.,Patriciu, V. CASSHH – Case Adaptive SSH Honeypot. Recent Trends in Computer Networks and Distributed Systems Security Volume 420 of the series Communications in Computer and Information Science pp 322-333. 2014
[15] Zakaria, WZ, Kiah, LM. Implementing a CBR Recommender for Honeypot Configuration using jCOLIBRI. Conference Paper. Conference: 3rd International Conference on Computer Science and Computational Mathematics (ICCSCM 2014). 2014
[16] Kácha, P.: IDEA: Classification of security events, their participants anddetection probes, in WSEAS TRANSACTIONS on COMPUTERS, pp. 213- 223, 2015.
[17] Modern Honey Network, 2015: http://threatstream.github.io/mhn/
[18] Biswas, J.: Analysis of Client Honeypots, in International Journal of Computer Science & Information Technologies, 5(4), 2014.
[19] A. G. Castillo, "Modelos y Plataformas de Agentes Software Móviles e Inteligentes para Gestión del Conocimiento en el Contexto de las Tecnologías de la Información", Tesis Doctoral dirigida por Luis Joyanes Aguilar, Facultad de Informática. Universidad Pontificia de Salamanca. Madrid, 2004.
[20] W. S. Humphrey, "The Personal Software Process℠ (PSP℠)", Software Engineering Intitute, CMU/SEI-2000-TR-022, ESC-TR-2000-022, 2000
[21] Carneige Mellon University,"Personal Software Process for Engineers, Using PSP0", 2006. Disponible: http://www.sei.cmu.edu.

Erik Michel Giraldo Giraldo nació en Manizales, Caldas, Colombia, el 4 de Noviembre de 1988. Ingeniero de Sistemas y Telecomunicaciones de la Universidad de Manizales. Candidato a Magister en Gestión y Desarrollo de Proyectos de Software de la Universidad Autónoma de Manizales.

Ha ejercido profesionalmente como Desarrollador Senior, Líder de Desarrollo de Software, Arquitecto de Soluciones, Coordinador de Desarrollo y Consultor de empresas de Telecomunicaciones como Claro Colombia, Tigo Colombia, Telintel.

Entre sus campos de interés se encuentra el Desarrollo de Software, Arquitectura de Software, Marcos de Referencias como ITIL, Cobit, PMBook, Metodologías Agiles de Desarrollo Saas, Paas.

Gustavo Isaza Echeverri Ph.D de la Universidad Pontificia de Salamanca (España) 2010, recibió su DEA/M.Sc en el 2008 en Ingeniería de Software, se graduó de Ingeniería de Sistemas y Computación en el año 1997 de la Universidad Autónoma de Manizales (Colombia), obtuvo su posgrado en desarrollo de software para redes en 1998 de la Universidad de los Andes (Colombia). Es Profesor Asociado e Investigador Asociado de la Universidad de Caldas, miembro del grupo de investigación GITIR, ha publicado más de 30 artículos y participado en conferencias relacionadas con inteligencia artificial aplicada a la Ciberseguridad, Bioinformática, AI en Video juegos, Computación distribuida y Paralela.