DEPARTAMENTO DE INFORMÁTICA Y AUTOMÁTICA

FACULTAD DE CIENCIAS

VNiVERSiDAD
D SALAMANCA

TESIS DOCTORAL

ARQUITECTURA MULTI-AGENTE ADAPTATIVA PARA LA DETECCIÓN DE ATAQUES EN ENTORNOS DINÁMICOS Y DISTRIBUIDOS

AUTOR

Cristian Iván Pinzón Trejos

DIRECTORES

Dr. D. Juan Manuel Corchado Rodríguez

Dr. D. Javier Bajo Pérez

2010

La memoria titulada “Arquitectura Multi-agente Adaptativa para la Detección de Ataques en Entornos Dinámicos y Distribuidos” que presenta Cristian Iván Pinzón Trejos para optar al Grado de Doctor en Informática y Automática por la Universidad de Salamanca ha sido realizado bajo la dirección del profesor Dr. Juan Manuel Corchado Rodríguez, profesor titular del Departamento de Informática y Automática, y el profesor Dr. Javier Bajo Pérez, profesor Encargado de Cátedra de la Universidad Pontificia de Salamanca.

Salamanca, septiembre de 2010

Los directores

El doctorando

Fdo: Dr. Juan Manuel Corchado Rodríguez
Profesor Titular de Universidad
Informática y Automática
Universidad de Salamanca

Fdo: Cristian I. Pinzón Trejos

Fdo: Dr. Javier Bajo Pérez
Profesor Encargado de Cátedra
Escuela Universitaria de Informática
Universidad Pontificia de Salamanca

Resumen

Las tecnologías de computación distribuidas surgieron dentro de las comunidades académicas y de investigación para satisfacer las necesidades de conexión y colaboración en estos sectores, pero poco a poco han pasado a formar parte del conglomerado de tecnologías de las empresas. En la actualidad existe una gran dependencia tanto en las empresas como en las comunidades de usuarios de los sistemas distribuidos. Sin embargo, al mismo tiempo que la tecnología de computación distribuida entra a formar parte de las aplicaciones empresariales, surgen preocupaciones acerca de la seguridad de la información. De esta forma, han aparecido amenazas principalmente orientadas a explotar vulnerabilidades en los componentes de las aplicaciones distribuidas. Estas amenazas afectan principalmente a la capa de aplicación ya que es considerada por los atacantes como un punto clave y susceptible a problemas de seguridad. Dos de las amenazas que han ganado relevancia en los últimos tiempos por la frecuencia de los ataques y su impacto negativo son los ataques de inyección SQL y los ataques de denegación de servicio basados en XML, en entornos de servicios Web (XDoS). Ambos ataques se caracterizan por su amplia variedad de técnicas de ataques, además de poner en riesgo tanto la confidencialidad e integridad de los datos y las aplicaciones, pero sobre todo la disponibilidad. Las medidas de seguridad existentes se centran primordialmente en garantizar la confiabilidad e integridad de los datos, sin prestar atención a la disponibilidad.

Es por ello que se hace necesario proponer nuevas soluciones de seguridad que hagan frente a este tipo de amenazas. En este trabajo se presenta la arquitectura AIDeMaS, una arquitectura multi-agente diseñada para detectar intrusiones en aplicaciones distribuidas. La arquitectura incorpora diferentes tipos de agentes para ejecutar cada una de las tareas del proceso de detección de ataques. El componente principal de la arquitectura AIDeMaS es un mecanismo de clasificación basado en un tipo de agente CBR-BDI, un tipo de agente deliberativo que integra un sistema de razonamiento basado en casos en su estructura interna. La detección de ataques de inyección SQL y ataques XDoS requiere un enfoque que pueda adaptarse a los constantes cambios en las técnicas de ataque y este tipo de agente resulta adecuado para ser aplicado en entornos altamente dinámicos. Finalmente, para determinar los patrones que corresponden a tipos de intrusiones, el agente CBR-BDI incorpora en su estructura interna técnicas de aprendizaje automático. El campo del aprendizaje automático se ha convertido en un campo prometedor en el ámbito de la detección de intrusiones y facilita el desarrollo de novedosas estrategias.

En resumen, la arquitectura propuesta en este trabajo presenta un avance significativo en el campo de la detección de intrusiones, mediante la aplicación de un conjunto de tecnologías y técnicas del campo de la Inteligencia Artificial.

Abstract

Distributed computing surged inside the academic and research communities trying to satisfy the growing need of connectivity and collaboration among the members of these communities, and have acquired a high importance for the industrial and business sectors. Nowadays there exists a high level of dependence of the business and users on the distributed systems. However, as the distributed computing becomes a relevant paradigm for the business applications, there are new problems related to the information security. In this way, it is possible to find different threats aimed at exploiting the vulnerabilities of the components of the distributed applications. These threats mainly affect the application layer of the systems, since this layer can be considered as a key point for the user access and it is sensitive to security problems. Two of the threats that have gained an increasing relevance during the last years, especially regarding the frequency of the attacks and the impact on the functioning of the systems, are the SQL injection attacks and the denial of service attacks based on XML, in web services environments (XDoS). Both types of attack are characterized by the wide variety of techniques that can be used for the attack, and are a risk for the confidentiality and integrity of the data and the applications, but mainly for the availability of the resources. Current security policies are focused on guarantee confidentiality and integrity of the data, but more efforts are required to guarantee availability of the resources

As it is necessary to provide new solutions to guarantee security for these types of threat, in this work is presented AIDeMaS, a multi-agent architecture designed for intrusion detection in distributed systems. The architecture defines different agent types that are specialized on the execution of the tasks that compose the attack detection process. The core component of the AIDeMaS architecture is a classification mechanism based on a CBR-BDI agent type, a deliberative agent type that integrates a case based reasoning engine in its internal structure. SQL injection and XDoS attacks detection requires new solutions and this study proposes a novel perspective where the detection strategy can be adapted to the continuous changes that occur in the techniques of attack, mainly based on the learning and adaptation capabilities of the CBR-BDI agents. Finally, to classify the attack patterns, the CBR-BDI agent incorporates automatic learning techniques in its internal structure. Machine learning is a promising field for the intrusion detection and allows proposing innovative strategies.

Summarizing, the proposed architecture represents a meaningful advance in the field of intrusion detection, providing a new perspective that makes use of a set of technologies and techniques of the Artificial Intelligence.

Agradecimientos

Durante estos años dedicados a la realización de esta tesis doctoral he conocido y compartido con personas que me han brindado todo el apoyo necesario para terminar con éxito este proyecto. A todas estas personas quiero expresarles mi agradecimiento.

Al Dr. Juan Manuel Corchado Rodríguez, quien me abrió las puertas del Grupo BISITE y me brindó toda la ayuda necesaria para que este proyecto personal y profesional fuera un éxito.

Al Dr. Javier Bajo Pérez, quien desde un principio me ha orientado y me ha hecho más llevadero todo el trabajo de investigación, indicándome las pautas necesarias para sacar el mayor provecho al tiempo disponible.

A Juan Francisco De Paz, sin su ayuda hubiese sido casi imposible sacar este proyecto en el tiempo disponible. A Sara Rodríguez, quien en todo momento me animó para terminar. También a Rosa Cano quien siempre me brindó su ayuda incondicional en los momentos difíciles, y todos los demás miembros del grupo BISITE.

A todos los compañeros de doctorado, con los que compartí muchas horas de trabajo en el Laboratorio de Tercer Ciclo. Siempre manteniendo un buen ambiente de trabajo e intercambiando buenas ideas.

Una mención muy especial a Yohanys, mi esposa, quién me ha acompañado en todo momento durante estos años lejos de casa. Sin su apoyo y compresión este proyecto no hubiese sido posible. A mis dos hijos, Cristian Jesús y Laura Patricia quienes son mi fuente de inspiración, quienes me motivan para seguir adelante y me dan la fuerza para afrontar las adversidades. A mi cuñada Elisa, por su ayuda invaluable. A mis padres, hermanos y demás familiares quienes en todo momento me han animado y dado la fortaleza necesaria para cumplir este sueño.

Finalmente, quiero extender un agradecimiento a SENACYT, al IFARHU y la Universidad Tecnología de Panamá, instituciones que confiaron en mi persona y aportaron los recursos necesarios para el desarrollo de este proyecto.

Índice

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1 Descripción del Problema

Desde la llegada de Internet a principios de los años 70, se ha producido un continuo crecimiento de nuevas aplicaciones que requieren un procesamiento distribuido. Factores como los avances en las redes de comunicación y la tecnología de hardware, la reducción de los costes en los equipos, y un mayor conocimiento del usuario final han contribuido a hacer de la computación distribuida una solución rentable, de alto rendimiento y tolerante a fallos (Belapurkar*, et al.*, 2009). Los sistemas distribuidos facilitan compartir recursos (CPU, memoria, almacenamiento, periféricos, etc.) y datos almacenados en bases de datos que generalmente se encuentran localizados en puntos remotos. Las áreas de aplicación de los sistemas distribuidos son muy amplias, incluyendo el sector financiero y comercial, el sector industrial, educación e investigación, salud, gobierno, etc. De esta forma, en la actualidad muchas personas desarrollan su vida diaria rodeada de elementos computacionales basados en las tecnologías distribuidas.

Sin embargo, si bien es cierto que la implementación de entornos basados en sistemas distribuidos proporciona incuestionables ventajas y beneficios, también es cierto que su implementación viene acompañada de cuestionamientos relacionados con la seguridad de las aplicaciones y los datos. El debate en torno a la seguridad se genera en base a las propias características inherentes de los sistemas distribuidos (Belapurkar*, et al.*, 2009). Una de las características claves es la participación de múltiples entidades en el sistema. Las entidades pueden ser usuarios o subsistemas que componen el sistema. En los sistemas distribuidos se parte de la confianza entre las entidades participantes para ejecutar las tareas, y es necesario introducir mecanismos que garanticen la confiabilidad. Sin embargo, si en las aplicaciones tradicionales resulta complicado disponer de mecanismos que garanticen la plena confiabilidad de los usuarios, en las aplicaciones distribuidas resulta mucho más difícil disponer de mecanismos de seguridad que garanticen una confianza plena en las entidades participantes. Una segunda característica a tener en cuenta es la heterogeneidad de las entidades involucradas. La heterogeneidad se basa en los tipos de sistemas

o usuarios, políticas, datos, recursos que los subsistemas consumen. Es el caso de Internet donde multitudes de sistemas, protocolos, políticas y entornos interactúan para crear una infraestructura escalable. Tan sólo es necesario que uno de los componentes en un subsistema tenga un fallo de seguridad para que toda o parte de la infraestructura sea vulnerable a ataques. Finalmente, la compartición de recursos y datos dentro de los entornos distribuidos es otra de las características que plantea riesgos de seguridad. El procesamiento de datos provenientes de fuentes no seguras abre la posibilidad de que se produzcan ataques contra las aplicaciones y los servicios habilitados, poniendo en riesgo la disponibilidad del sistema.

En la actualidad, se dispone de un sinnúmero de mecanismos de seguridad para proteger las aplicaciones y los datos. Estos mecanismos incluyen cortafuegos, filtros, sistemas de autenticación, encriptación de la comunicación, sistemas de detección de intrusiones, auditorias y monitorización, sistemas trampas (*honeypots*), señales de seguridad (*security tokens*), dispositivos biométricos, mecanismos espías (*sniffers*), análisis de seguridad a nivel de archivos, zona desmilitarizada, etc. Generalmente estos mecanismos se centran en garantizar la confidencialidad, integridad y privacidad de los recursos y los datos. Sin embargo, estos mecanismos de seguridad prestan poca o ninguna atención a garantizar la disponibilidad de los datos y el funcionamiento fiable de las aplicaciones y los servicios. La disponibilidad garantiza proveer, de manera fiable, los servicios y los datos a los usuarios cuando estos los soliciten frente a interrupciones intencionales o accidentales. La no disponibilidad de los datos y los servicios cuando son requeridos por los usuarios autorizados provoca inevitablemente a las organizaciones daños económicos y pérdida de confianza. Esta situación exige la necesidad de disponer de nuevos mecanismos que puedan afrontar las amenazas que ponen en riesgo la disponibilidad de los datos, las aplicaciones y los servicios. En este trabajo de investigación se estudian dos ataques de seguridad, inyección SQL y ataques XDoS, que representan una amenaza para los entornos emergentes que se desarrollan en base a los sistemas distribuidos. Estos ataques se caracterizan por una alta dinamicidad, evolucionando continuamente mediante distintas estrategias de ataque, lo que hace difícil su detección mediante los mecanismos de seguridad disponibles hoy en día.

En este trabajo se abordan los ataques más frecuentes, severos y dinámicos a nivel de la capa de aplicación en sistemas distribuidos y se proponen soluciones innovadoras. En concreto se estudian los ataques de inyección SQL (Anley, 2002) (Litchfield*, et al.*, 2005), (Halfond*, et al.*, 2006), ejecutados contra las aplicaciones dinámicas que gestionan una base de datos, y los ataques XDoS (Im y Song, 2005) (Moradian y Håkansson, 2006), (Vorobiev y Han, 2006), (Gruschka y Luttenberger, 2006), ejecutados contra entornos de servicios Web. Los ataques de inyección SQL, consisten en la inyección de código SQL sobre las consultas SQL enviadas a la base de datos. Este tipo de ataque es bastante conocido y estudiado pero en la actualidad sigue siendo uno de los ataques más frecuente y

resulta difícil de detectar debido a su amplia variedad de técnicas de ataque y continua evolución. Los ataques XDoS son una variante de los ataques de denegación de servicio tradicionales, pero se diferencian principalmente en la explotación del lenguaje XML utilizado para construir las peticiones de servicios Web. Existen diferentes técnicas de ataque XDoS, como las inyecciones de código XML, anidamiento XML complejo, peticiones XML demasiado grandes, envío masivo de peticiones XML, etc. El objetivo de los ataques de inyección SQL y XDoS es principalmente afectar a la disponibilidad de las aplicaciones, los datos y los servicios disponibles, inhabilitando el acceso a los usuarios autorizados.

En este trabajo de tesis doctoral se propone AIDeMaS, una arquitectura multi-agente adaptativa, como solución innovadora al problema de la detección de los ataques de inyección SQL y XDoS en entornos distribuidos y dinámicos. La arquitectura AIDeMaS se basa en la teoría de agentes, ya que los sistemas multi-agente han ganado importancia para resolver problemas en entornos distribuidos altamente dinámicos durante los últimos años gracias a las capacidades inherentes que poseen los agentes. Los agentes se caracterizan por sus capacidades, tales como autonomía, reactividad, pro-actividad, habilidades sociales, razonamiento, aprendizaje y movilidad entre otras (Wooldridge y Jennings, 1995), (Wooldridge, 2002). Cuando los agentes se organizan dentro de un sistema y se cumplen ciertas condiciones, se habla de un sistema multi-agente para desarrollar las tareas, y la resolución del problema se plantea desde un enfoque distribuido (Mas, 2005).

La arquitectura AIDeMaS presenta como principal característica la utilización de agentes deliberativos, concretamente agentes deliberativos basados en el modelo BDI (*Beliefs, Desires, Intentions)* (Georgeff y Lansky, 1987), (Bratman*, et al.*, 1988), (Pokahr*, et al.*, 2003). En el modelo BDI la estructura interna de un agente y su capacidad de elección se basan en aptitudes mentales. Esto tiene la ventaja de utilizar un modelo natural (humano) y de alto nivel de abstracción. El modelo BDI utiliza "*Beliefs*" como aptitudes informacionales, "*Desires*" como aptitudes motivacionales e "*Intentions*" como aptitudes deliberativas de los agentes. Sin embargo, debido a la dinámica en la evolución de los ataques bajo estudio, se requiere incrementar la capacidad que poseen los agentes con una arquitectura deliberativa BDI pura. Las capacidades de los agentes se pueden modelar de distintas formas y con diferentes metodologías (Wooldridge y Jennings, 1995). Una de las alternativas es la utilización de sistemas de razonamiento basado en casos (CBR). Un sistema de razonamiento basado en casos permite la resolución de nuevos problemas a través de experiencia ganada con resoluciones de problemas anteriores (Aamodt y Plaza, 1994). Incorporando un modelo de razonamiento basado en casos en la estructura interna de los agentes, es posible diseñar agentes con una mayor capacidad de adaptación a entornos dinámicos y cambiantes. Se trata de un comportamiento deseable para la detección y prevención de ataques de inyección SQL y XDoS. A los agentes deliberativos BDI que hacen uso de sistemas CBR se les llama agentes CBR-BDI (Bajo*, et al.*, 2010), (Pinzón*, et al.*, 2009b),

(Pinzón*, et al.*, 2008a), (Carrascosa*, et al.*, 2008), (Corchado*, et al.*, 2003). El agente CBR-BDI razona a partir del conocimiento almacenado en una memoria de casos que contiene experiencias previas. Cuando un agente CBR-BDI se encuentra con un nuevo problema, el agente utiliza los casos almacenados en la memoria de casos más similares al caso bajo estudio, para generar una nueva solución. En resumen, para el caso que nos interesa, los agentes CBR-BDI poseen un mayor grado de autonomía, una mayor capacidad para la resolución del problema de la detección, una mejor adaptación a la evolución de los ataques, y facilitan el aprendizaje al identificar posibles soluciones basándose en experiencias pasadas.

Finalmente, la nueva estrategia de detección diseñada en la arquitectura AIDeMaS incorpora técnicas y algoritmos de aprendizaje automático. La identificación de intrusiones representa una de las tareas más complicadas en la detección de intrusiones. Los dos métodos de detección más conocidos son los basados en uso indebido (*misuse deteccion*), y detección basada en anomalía (*anomaly deteccion*). La detección por uso indebido se basa en la identificación de intrusiones mediante la comparación de firmas o patrones de ataque. Por el contrario, la detección basada en anomalía permite detectar cualquier desviación del comportamiento considerado normal. La detección de intrusiones utilizando técnicas y algoritmos del aprendizaje automático se he convertido en un campo de investigación que ofrece perspectivas prometedoras de cara a la resolución de los problemas presentes en las técnicas tradicionales (Wu y Banzhaf, 2010), (Kandeeban y Rajesh, 2010). Es el caso del enorme volumen de tráfico generado en las aplicaciones distribuidas, y la continua evolución de las técnicas de ataque. En el aprendizaje automático, los sistemas incrementan su conocimiento o sus habilidades para cumplir una tarea. El sistema aplica inferencias a una determinada información para construir una representación apropiada de algún aspecto relevante de la realidad o de algún proceso(Sierra, 2006). Dependiendo del tipo de estrategia, la disponibilidad de un conjunto de entrenamiento, y la ayuda que reciba el sistema, es posible diferenciar en dos métodos el tipo de aprendizajes utilizado: método supervisado, y no supervisado. En la detección de intrusiones, los métodos supervisados son frecuentemente aplicados en la detección de usos indebidos y los métodos no supervisados en la detección por anomalía. Ambas estrategias presentan sus ventajas y desventajas. No obstante, la estrategia basada en anomalía se presenta especialmente prometedora para hacer frente a la continua evolución de los ataques. En el caso de la arquitectura AIDeMaS, los agentes CBR-BDI diseñados incorporan en su estructura interna técnicas de detección basadas en anomalía. Esta estrategia permite incrementar la capacidad de los agentes CBR-BDI dotándoles de una mayor capacidad de razonamiento y aprendizaje. Como resultado, se puede alcanzar una mayor efectividad a la hora de identificar y bloquear las intrusiones.

Para la evaluación de la arquitectura AIDeMaS se propone la aplicación de un caso de estudio para cada uno de los ataques estudiados. En el primer caso de estudio se evalúa la efectividad de la arquitectura AIDeMaS frente a los ataques

de inyección SQL teniendo como escenario una aplicación multi-agente instalada en una residencia geriátrica. El segundo caso de estudio evalúa la efectividad de la arquitectura AIDeMaS ante los ataques XDoS, utilizando para ello una aplicación de servicios Web creada previamente.

1.2 Motivación e Hipótesis

Las tecnologías de computación distribuidas están transformando el mundo, que hoy conocemos, de manera significativa. Estos sistemas permiten la interacción de sistemas heterogéneos a través de una plataforma de comunicación común. Esta forma de trabajar ofrece grandes ventajas y beneficios para facilitar la participación de distintas entidades, lograr una mayor interoperabilidad entre las aplicaciones heterogéneas, y facilitar la compartición de recursos y datos, etc. Garantizar la seguridad de las aplicaciones distribuidas resulta bastante complejo y las amenazas que afectan a estos sistemas van en aumento. Los mecanismos de seguridad existentes resultan poco eficaces debido a su incapacidad para adaptarse a los cambios de comportamiento en los ataques, exponiendo los datos y las aplicaciones a los intrusos con cierta facilidad. Dos de los ataques más peligrosos son los ataques de inyección SQL y los ataques XDoS. Estas amenazas a la seguridad se caracterizan por disponer de distintas técnicas de ataque y una facilidad para evolucionar y eludir las medidas de seguridad configuradas. Ambos ataques ponen en riesgo la disponibilidad de los datos, las aplicaciones y los servicios en el caso de los servicios Web.

En este trabajo de tesis doctoral, el objetivo final es el de diseñar una arquitectura multi-agente que permita construir sistemas multi-agente para detectar y bloquear intrusiones en aplicaciones distribuidas. Para realizar esta función, la arquitectura debe proporcionar los componentes necesarios para realizar el proceso de detección de una manera innovadora y eficiente. Así pues, se ha diseñado en su estructura interna un mecanismo de detección con capacidad de adaptación para evolucionar a los cambios de comportamiento de los ataques. Esto requiere disponer de agentes con capacidades avanzadas de razonamiento y aprendizaje. Además, la arquitectura debe facilitar la distribución de las tareas en el proceso de detección de intrusiones y disponer de una estrategia para continuar funcionando bajo condiciones adversas. En este sentido, la arquitectura se plantea utilizando un diseño jerárquico en capas que favorece la distribución de las tareas entre los distintos agentes, al mismo tiempo que facilita el funcionamiento continúo de la arquitectura limitando el impacto de errores y facilitando su recuperación. Finalmente, se debe facilitar la interacción entre la arquitectura y el experto en seguridad en cualquier momento y lugar. En esta dirección, la arquitectura se plantea para manejar una interfaz en dispositivos móviles, eliminando las restricciones de tiempo y lugar.

Para alcanzar el objetivo propuesto, es necesario realizar un minucioso estudio del arte de las técnicas existentes utilizadas en la detección de intrusiones, de la tecnología de agente y sistemas multi-agente, sistemas de razonamientos basados en casos, y técnicas y algoritmos del campo del aprendizaje automático. Los objetivos específicos de este trabajo de investigación son los siguientes:

- Diseñar una arquitectura multi-agente que permita construir sistemas multi-agente para detectar los ataques estudiados y en diversos escenarios reales. Para el diseño de la arquitectura se plantea una revisión de las diversas propuestas existentes. Se analizarán las ventajas y desventajas de las soluciones existentes, de cara a considerar los aspectos positivos, mejorar los aspectos negativos, y proponer una nueva estrategia en el campo de la detección de intrusiones.

- Diseñar y desarrollar un mecanismo de detección de intrusión inteligente y adaptable como núcleo de la arquitectura multi-agente propuesta. Este mecanismo de detección debe tener la capacidad de evolucionar y mantener la efectividad para detectar nuevos ataques y variaciones en los patrones de ataque estudiados. Para lograr un mecanismo de detección de este tipo, se estudiaran técnicas y algoritmos de Inteligencia Artificial que incluyen el modelo BDI para los agentes, sistemas CBR, y técnicas del aprendizaje automático. Se propondrán nuevos algoritmos que permitan una integración de estas técnicas dentro del mecanismo de detección.

- Desarrollar y evaluar un prototipo cercano a la implementación del sistema teniendo como base el diseño de la arquitectura multi-agente propuesta, incorporando el mecanismo de detección. El prototipo del sistema multi-agente, más concretamente el mecanismo de detección, será probado en más de un escenario real para evaluar la efectividad en la detección y bloqueo de los ataques estudiados.

- Evaluar empíricamente la aplicabilidad de la tecnología de agentes en la detección de intrusiones en entornos distribuidos altamente dinámicos reales.

1.3 Metodología de Investigación

La metodología de la investigación aplicada en este trabajo de tesis doctoral se ha centrado en el modelo de Investigación-Acción (*Action-Research*). Este modelo se caracteriza por un proceso continuo, similar a una espiral, donde se presenta el problema, se lleva a cabo un diagnostico, se diseña una propuesta

como solución, se aplica y finalmente se evalúa, para nuevamente iniciar el proceso partiendo de un nuevo problema. En el caso concreto de este trabajo, el proceso se dividió en 5 actividades importantes. Las actividades son las siguientes:

- Identificación y planteamiento del problema: Se requiere identificar el problema para limitar su alcance, plantear una hipótesis y establecer los objetivos. En este trabajo, la descripción del problema comprende la detección de intrusiones en los entornos de las aplicaciones distribuidas, más concretamente los ataques de inyección SQL y los ataques XDoS, apuntados a poner en riesgo la disponibilidad de los datos, las aplicaciones y los servicios.

- Revisión del estado del arte: Dentro de esta actividad se realiza una revisión de las tecnologías disponibles y las propuestas existentes, para analizar el problema, conocer lo que está hecho y proponer una nueva solución. La revisión del estado del arte realizada en este trabajo comprende la tecnología de agentes, sistemas CBR, técnicas de aprendizaje automático, y principalmente las propuestas existentes como solución al problema de la detección de los ataques estudiados. La forma más práctica de resumir los detalles importantes del estado del arte es utilizando el recurso de las tablas comparativas.

- Definición de la propuesta: En esta actividad se plantea la nueva solución partiendo de una investigación preliminar. La nueva solución debe corresponder con los objetivos propuestos. En este caso, se propone el diseño de una arquitectura multi-agente jerárquica incorporando un mecanismo de detección de intrusiones adaptativo basados en agentes CBR-BDI, y la utilización de técnicas del aprendizaje automático para resolver la detección de las intrusiones de una forma eficaz.

- Implementación y evaluación de la solución en entornos reales: Se realiza una implementación del prototipo de la propuesta, en un entorno real, para llevar a cabo una evaluación. La arquitectura propuesta se ha implantado en dos escenarios reales para abordar los ataques de inyección SQL y los ataques XDoS. A partir de la implementación se ha podido obtener los resultados que han permitido valorar la efectividad de la arquitectura en la detección de las intrusiones estudiadas. Además de facilitar una comparación teórica, por sus características, con las propuestas existentes.

- Publicación de los resultados: Con los resultados obtenidos y su evaluación, se inicia el proceso de divulgación de los resultados para someterlos a la valoración de la comunidad científica. En esta dirección se han publicados artículos en congresos internacionales afines a la investigación, así como en revistas de impacto en el tema. Esta actividad ha favorecido un importante proceso de retroalimentación a lo largo de la investigación.

1.4 Estructura de la Memoria

La memoria de este trabajo ha sido estructurada en 6 capítulos como se detalla a continuación.

- El capítulo 1 constituye el planteamiento de la tesis. En este capítulo se ha realizado la descripción del problema a resolver. Se han establecido los objetivos que se quiere alcanzar y la hipótesis basada en una propuesta de una arquitectura multi-agente adaptativa como solución al problema. Adicionalmente, se han indicado los motivos que han originado la investigación y finalmente se concluye con la metodología utilizada a lo largo de la investigación.

- El capítulo 2 comprende una parte de la revisión del estado del arte. Concretamente corresponde a la revisión de la seguridad en los entornos dinámicos distribuidos, identificando los problemas por resolver. Se estudian los conceptos de detección de intrusiones, la clasificación de los sistemas de detección de intrusiones desde diferentes puntos de vistas; se analizan las ventajas y desventajas de los sistemas de detección de intrusiones, y se evalúan las propuestas de arquitecturas IDS basadas en agentes. Finalmente, se hace un estudio detallado de los ataques de inyección SQL y los ataques XDoS.

- El capítulo 3 completa la revisión del estado del arte realizando una revisión de la tecnología de agentes y sistemas multi-agente, los sistemas de razonamiento basados en casos y técnicas del campo de la minería de datos y el aprendizaje automático. En el tema de los agentes se estudian los conceptos de agentes y sistemas multi-agente, la comunicación entre los agentes. Se revisan las arquitecturas internas de los agentes especialmente la arquitectura deliberativa BDI. En relación a los sistemas CBR, se estudia el concepto de CBR, el ciclo de vida de un sistema CBR y la memoria de casos. Completando con un análisis de las ventajas y desventajas de la aplicación de los sistemas CBR. Finalmente, se termina el capítulo con una revisión de diferentes técnicas del campo de la minería de datos y el aprendizaje automático, especialmente las redes neuronales y los árboles de decisiones aplicados en este trabajo.

- El capítulo 4 constituye la arquitectura propuesta. En este capítulo se hace la descripción de la arquitectura AIDeMaS, justificando el modelo jerárquico en capas utilizado en el diseño de la arquitectura. Se continúa con la explicación de la arquitectura interna de los distintos tipos de agentes diseñados, la arquitectura funcional y la arquitectura multi-agente. Adicionalmente, se explica cómo se lleva a cabo la comunicación interna de los agentes en las diferentes capas de la arquitectura. Finalmente, se explica el mecanismo de

clasificación en dos fases incorporado en la arquitectura AIDeMaS. Se describe de manera detallada la estructura interna de los agentes CBR-BDI; especialmente las distintas fases del ciclo CBR donde se incorpora una técnica del aprendizaje automático para la clasificación de las peticiones de usuarios.

- El capítulo 5 se centra en los dos casos de estudios aplicados para evaluar la arquitectura AIDeMaS. El primer caso de estudio está enfocado en los ataques de inyección SQL utilizando como escenario real una aplicación multi-agente instalada en una residencia geriátrica. El segundo caso de estudio corresponde a los ataques XDoS en entornos de servicios Web, y para ello se ha utilizado una aplicación de servicios Web previamente construida.

- El capítulo 6 presenta las conclusiones obtenidas y el trabajo futuro propuesto. Se plantea una comparativa, a nivel teórico, de las propuestas existentes con la arquitectura AIDeMaS. Se indican algunas de las posibles líneas de investigación futura, y finalmente se proporciona una lista de las referencias bibliográficas utilizadas para llevar a cabo este trabajo.

Capítulo 2

Seguridad en Entornos Dinámicos y Distribuidos

Introducción

Las tecnologías de la información y la comunicación que han transformado Internet tal y como lo conocemos hoy en día, están íntimamente relacionadas con las tecnologías de computación distribuida. Nuevos modelos de interacción están evolucionando con el surgimiento de las recientes tecnologías para colaborar y compartir datos. Un ejemplo claro se está dando a nivel de las empresas y organizaciones, las cuales están siendo testigos del incremento en la colaboración y la compartición de datos entre diferentes entidades, trayendo consigo la necesidad de adaptar las estructuras para ejecutar procesos y compartir recursos de una manera distribuida.

No obstante, proporcionar seguridad en entornos distribuidos representa un enorme desafio. La computación distribuida busca alcanzar una integración completa de los sistema de computación heterogénos y los recursos de datos para ofrecer una espacio de computación global. Para alcanzar este objetivo se requieren cambios revolucionarios en el campo de la computación y más concretamente en el aspecto de la seguridad. ¿La principal razón? se requiere compartir recursos a través de las redes de comunicación. Los desafios en los sistemas distribuidos provienen de las amenazas de seguridad, las cuales explotan deficiencias en los protocolos asi como del propio sistema operativo, y pueden extenderse a ataques a las aplicciones, bases de datos, etc. Las amenazas incluyen ataques de denegación de servicio, virus, ataques de desbordamiento de memoria, gusanos, etc., los cuales causan graves daños economicos, de reputación entre otros.

En este capítulo se hace una identificación de los requisitos de seguridad en entornos de dinámicos y distribuidos y se revisan los tipos de amenazas en los diferentes niveles de un sistema distribuido. Se centrará la atención en dos tipos de amenaza, los ataques de inyección SQL y los ataques de denegación de servicio basados en XML (*XDoS*), los cuáles ponen en riesgo la disponibilidad de

los servicios y los recursos a los usuarios consumidores. Evaluaremos las diferentes soluciones propuestas en la literatura.

2.1 Seguridad en Entornos Dinámicos y Distribuidos

La seguridad de la información se basa en tres principios fundamentales: Confidencialidad, Integridad y Disponibilidad, más conocidos como "*The Big Three*" (C.I.A) (Krutz y Vines, 2002). En función de la aplicación y el contexto, cada principio puede jugar un papel más importante que el resto. No obstante, debe quedar claro que todos los controles y las medidas de seguridad, todas las amenazas y vulnerabilidades, y los procesos de seguridad son medidos desde el punto de vista de estos tres principios. La Figura 2.1 presenta de forma gráfica los 3 principios fundamentales.

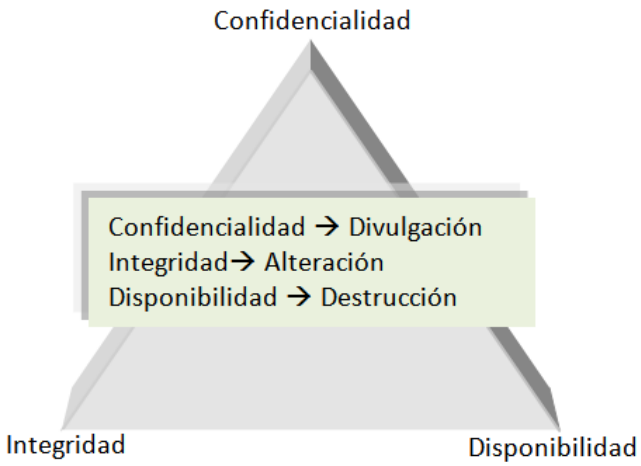

Figura 2.1. El gran árbol: confidencialidad, integridad y disponibilidad

- Confidencialidad: Previene la divulgación de información no autorizada, ya sea de forma intencional o accidental. La criptografía y el control de acceso (Konheim, 2007), (Stamp, 2006) son dos de los mecanismos aplicados para garantizar la confidencialidad.

- Integridad: Asegura que la información es completa, exacta y auténtica. Con la integridad se previene que la información sea modificada por usuarios no autorizados, e inclusive se previene la modificación intencional o accidental

por usuarios autorizados. Además, la integridad debe asegurar la consistencia de los datos tanto dentro como fuera de los límites de la organización.

- Disponibilidad: Es siempre requerida pero no siempre es considerada para formar parte de la seguridad. Asegura que los usuarios autorizados tengan acceso a la información y a los recursos de manera oportuna y libre de interrupciones. Es evidente que si un sistema no está disponible, entonces tanto la integridad como la confidencial pasan a un segundo plano (Dunsmore y Brown, 2000). La redundancia, la tolerancia a fallo y los sistemas de respaldo son algunos de los mecanismos aplicados para garantizar la disponibilidad. Los ataques de Denegación de Servicio (DoS) se han convertido en la primera amenaza potencial para afectar la disponibilidad de la información y los recursos en los sistemas (Carl*, et al.*, 2006).

Además de estos tres principios, existen otros elementos claves asociados a la seguridad se deben considerar:

- Autenticación: Se refiere al proceso de verificar que el individuo es ciertamente quien dice ser. Este concepto es lo más fundamental en la seguridad. Ejemplos: PIN, nombre de usuario y clave, licencia de conducir, Pasaporte, etc.

- Autorización: Es un paso fundamental, requerido antes de permitir a cualquier persona/entidad llevar a cabo una operación en el sistema o acceder a cualquiera parte el sistema. Un requisito obligatorio en seguridad es el establecimiento de diferentes niveles de acceso a diferentes partes de una operación o de un sistema informático. El tipo de acceso se establece por la identidad de la persona/entidad que requiere el acceso y el tipo de operación o parte del sistema al que necesita acceder.

- No repudio: Servicio de seguridad que asegura que el origen de una información no puede rechazar su transmisión o su contenido, y/o que el receptor de una información no puede negar su recepción o su contenido. En general, los protocolos de no-repudio en el mundo de la seguridad garantizan que dos partes no puedan negar que existe una interacción entre ellas. Sin embargo, los protocolos de no-repudio son deseables en la teoría, pero su implementación termina siendo muy costosa y son muy poco usados en la práctica.

- Auditoría: El objetivo es garantizar al encargado de la seguridad la capacidad de determinar quien está realizando una actividad no permitida o identificar quién ha generado algún tipo de error. Esto permite rastrear y asignar responsabilidades cuando se detecta un tipo de incidente malicioso. El mecanismo más frecuente para ejecutar este tipo de auditoría es apoyarse en

medios de autenticación que lleven a cabo un sistema de registros de entradas con todas las acciones realizadas por cada usuario autenticado.

Desde el punto de vista de los sistemas distribuidos, los problemas de seguridad tienen que ser analizados considerando las diferentes capas y componentes que intervienen en una arquitectura distribuida. La Figura 2.2 presenta una vista de las capas que podemos encontrar en una estructura de tecnología de información empresarial genérica.

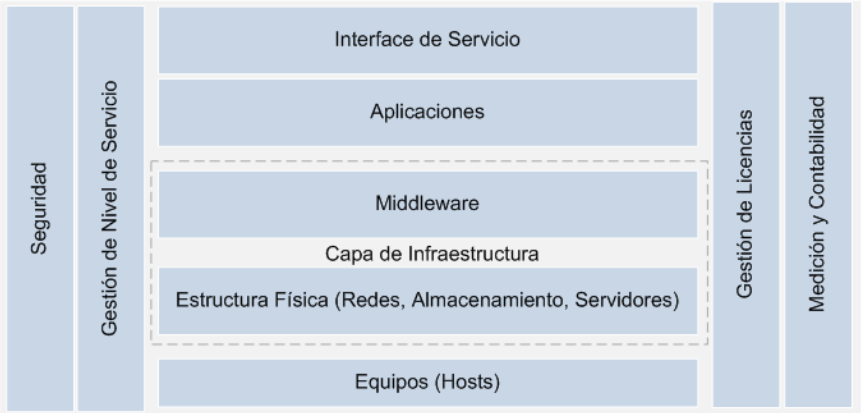

Figura 2.2. Vista en capas de la infraestructura de tecnología de información empresarial

La vista presenta los 4 componentes principales: Equipos, infraestructura, aplicaciones y servicios.

- Equipos: Consiste de los equipos de escritorio, servidores, etc. que forman la capa más baja en la arquitectura tecnológica.

- Infraestructura: consiste en la infraestructura de red, mecanismos de almacenamiento, y software de conectividad (*middleware*) usados por las aplicaciones, equipos y la capa de servicio.

- Las aplicaciones: Son las aplicaciones comerciales personalizadas usadas en las empresas en el día a día.

- Servicios: Corresponde a los servicios basados en estándares puestos a disposición de los usuarios externos y usuarios interno de la empresa.

Cada capa de la arquitectura requiere una especial atención para proporcionar el nivel de seguridad requerido. Más allá de los desafíos existentes para proporcionar los mecanismos de autenticación adecuados para controlar el acceso de un volumen enorme de usuarios, los niveles de confidencialidad e integridad requeridos para la compartición de los datos; los entornos

distribuidos presentan otros grandes desafíos que incluyen: políticas de integración, autorización y asuntos relacionados a la gestión de credenciales, todo debido principalmente a la naturaleza heterogénea de los sistemas (Chakrabarti*, et al.*, 2008).

- Políticas de Integración: Un sistema distribuido está caracterizado por la heterogeneidad no solo a nivel de recursos sino también a nivel de políticas. Una de las maneras que los sistemas pueden abordar el tema de la interoperabilidad es a través de la estandarización.

- Autorización: Los sistemas de autorización deben ser interoperables a través de múltiples sistemas teniendo políticas heterogéneas.

- Gestión de credenciales: Las credenciales van más allá de identificar un usuario. Las credenciales pueden autorizar a un usuario a acceder a ciertos recursos o pueden ser usados como pruebas de autenticación. En entornos heterogéneos, la gestión de credenciales es un tema importante. Se requiere investigar mecanismos para almacenar, acceder y gestionar credenciales en entornos de computación distribuida.

Además de estos desafíos, otros aspectos válidos en la actualidad deben ser considerados cuando se plantea el uso de sistemas distribuidos; a continuación son mencionados. Finalmente, una recopilación de las problemas de seguridad más populares y que requieren igual atención son resumidos en (Bruce y Dempsey, 1997).

Otros aspectos de seguridad a considerar incluye:

- Es importante reconocer la complejidad que se deriva de un entorno de computación en crecimiento donde el número de sistemas que demandan seguridad puede convertir la tarea en un proceso costoso y complicado. Se requiere establecer claramente la diferencia entre la seguridad aplicada a un sistema centralizado y la que demanda un sistema distribuido.

- En un sistema distribuido se interactúa con vecinos y extraños de manera muy frecuente, creando relaciones de confianza con los vecinos no así con los extraños. Es necesario reconocer claramente cuándo se está interactuando con un vecino y cuándo se está haciendo con un extraño que puede estar apuntando un ataque hacia nuestro sistema.

- Es importante identificar todas las fuentes de entrada de amenazas, en el caso de los virus las propias personas o empleados puede introducir estas amenazas al utilizar diferentes dispositivos de almacenamiento sin una revisión previa exhaustiva. El costo que involucrada limpiar completamente de virus un sistema es realmente alto.

- ¿Cómo saber cuál es el costo que involucra la protección de los activos de las amenazas? En este sentido se hace necesario un análisis de riesgo para alcanzar un sistema de seguridad balanceado. Uno de los grandes problemas de procesamientos distribuidos es la dificultad de decidir qué activo es valioso y cuál no lo es, y donde está localizado cada uno. Es importante ofrecer un coste de protección acorde al valor de los activos y para ello se debe entender primero qué se va a proteger antes de protegerlo. Adicional, no se puede considerar la ausencia de problemas de seguridad con la ausencia de riesgos.

- La clasificación de los datos dentro de categorías es importante a la hora de compartir información considerando los requerimientos de confidencialidad y disponibilidad. Sin un sistema de clasificación, mecanismos de protección restrictivos pueden ser aplicados a cualquier dato, limitando potencialmente la productividad. O en caso contrario, unas medidas de seguridad demasiados flexibles podría incrementar el riesgo. La seguridad y la disponibilidad de los datos determinarán que medidas de seguridad son requeridas y cómo estas medidas deben ser aplicadas.

- Es importante una política robusta para el manejo de la información del proceso de inicio de sesión (usuario/clave) en los diferentes sistemas. Frecuentemente se pueden encontrar entornos distribuidos donde se requiere hacer el proceso de inicio de sesión en diversos sistemas cada uno requiriendo datos de sesión diferentes (usuario/clave). Exponer esta información en lugares visible pone en riesgo la seguridad del sistema a pesar de la robustez del mecanismo de autenticación y autorización.

- El acceso remoto a la red corporativa y a los sistemas es un tema no sólo de seguridad sino también de políticas. Muchos empleados tienen habilitado el acceso a los sistemas desde sus casas, teniendo habilitado una puerta de entrada a la red corporativa. Esto puede invitar a que otras personas no autorizadas intenten utilizar estos accesos para entrar a los sistemas. Si bien es cierto existen muchas soluciones técnicas para garantizar los niveles de seguridad requeridos, para garantizar el acceso sólo a los usuarios autorizados, también es cierto que se requiere un control estricto con la conexión habilidad en el equipo informático de uso doméstico.

Problemas más relevantes que encaran las organizaciones:

- Alcanzar el balance correcto: Debe existir un balance entre necesidades de seguridad y otras necesidades consideradas prioritarias para alcanzar una utilización efectiva del sistema. Se tienen que asignar los recursos necesarios para una protección acorde al valor de los activos que se requieren proteger. También es importante evaluar aspectos de desempeño con respecto a los requerimientos de seguridad.

- Débil autenticación: Es importante asegurar que las personas o los sistemas puedan demostrar que son quienes dicen ser. Adicional, un sistema robusto de autenticación previene que la información de inicio de sesión en los sistemas pueda ser descubierta mientras viaja en el tráfico de la red.

- Herramientas de administración: Herramientas para procesos de autenticación, control de acceso, audito y detección son necesarias en los entornos distribuidos. Sin embargo las herramientas deben soportar una variedad de productos así como trabajar en una diversidad de entornos.

- Internet: Serias consideraciones se deben tomar en cuenta con el acceso a la Internet. Es necesario plantear mecanismos que garanticen que la seguridad no ha sido comprometida por el hecho de mantener una conexión a Internet.

- Puntos débiles en la red: Considerar todos los puntos que pueden convertirse en puntos vulnerables en la red. Un sistema vulnerable se traduce en la puerta de acceso a sistemas y aplicaciones más críticas. Se requiere garantizar que todos los sistemas reúnan estándares definidos y un audito global del sistema. Aquellos sistemas vulnerables deben ser aislados de la red corporativa.

- Diversas tecnologías: Todavía existe una escasez de estándares que evita la adopción de soluciones completas para solucionar los problemas. Existen muchas tecnologías eficientes y robustas para direccionar casi todos los problemas de seguridad, sin embargo, la interacción entre estas tecnologías se hace difícil a la hora de enlazarlos.

- Acceso físico: Controlar el acceso físico a los sistemas por personas no autorizadas es una prioridad. Gran parte de los incidentes de seguridad reportados son responsabilizados a empleados quienes no han seguido las políticas establecidas para la protección del sistema.

- Políticas y Procedimientos inapropiados: La implementación de computación distribuida requiere un planteamiento de nuevas políticas y procedimientos acordes a los cambios en el entorno y la forma de realizar el trabajo.

- Educación: Los administradores de sistemas requieren una actualización de sus conocimientos desde el punto de vista de la administración de un sistema distribuido. Muchos ataques son realizados por el desconocimiento dentro de la organización de puntos vulnerables.

- Falta de Planes: La organización requiere planes actualizados para hacer frente a los cambios introducidos por un sistema distribuido. La falta de planes puede producir fallos en implementaciones dejando puertas abiertas para ser explotadas por usuarios maliciosos.

La seguridad en los sistemas distribuidos es crítica y absolutamente esencial; no obstante, también resulta un gran desafío (Belapurkar, *et al.*, 2009). Los sistemas distribuidos poseen características que determinan su naturaleza (múltiples entidades, heterogeneidad, concurrencia y compartición de recursos). Adicional, hay que considerar las características deseables (entornos abiertos, escalables y transparentes). Con estas características, la seguridad se convierta en una tarea demasiado compleja. Para alcanzar un sistema distribuido seguro en futuras redes y aplicaciones, los aspectos de seguridad que incluyen confidencialidad e integridad de los datos, auntenticación, no repudio, privacidad, control de acceso y disponibilidad se deben direccionar completamente.

Finalmente, es importante señalar el giro que está tomando la seguridad para adaptarse a las nuevas demandas de protección. La seguridad se está moviendo a las capas de alto nivel, a la protección de los periféricos, protección de las identidades, estandarización e integración de políticas heterogéneas e infraestructura.

2.1.1 Amenazas Comunes dentro de los Entornos Distribuidos

Tomando en cuenta cada unas de las capas de una arquitectura distribuida, los problemas de seguridad requieren ser analizados desde esta perspectiva. Cada capa puede estar compuesta por un conjunto de componentes que pueden añadir puntos vulnerables, exponiendo la aplicación a tipos diferentes de amenazas. A continuación resumimos las amenazas para cada una de las capas de un sistema distribuido tomando como referencia la recopilación presentada en (Belapurkar, *et al.*, 2009).

- A nivel de Host las amenazas aumentan debido a la transferencia de código entre sistemas o vulnerabilidades en programas de software instalados en los equipos y que pueden ser explotados. Las amenazas a nivel de los equipos caen en dos grandes categorías: basado en código transitorio (software malicioso, programas espías, ataques de denegación de servicio). Basado en código residente (desbordamiento de memoria, ataques para escalar privilegios, ataques de inyección). Todas las amenazas bajo esta categoría deben ser tratadas como de impacto alto y de relevancia inmediata.

- A nivel de infraestructura: Las amenazas son encerradas en 3 grandes categorías: Amenazas en las redes (ataque de denegación de servicios, amenazas DNS, ataques de enrutamiento, amenazas en redes (inalámbricas) de alta velocidad, amenazas en redes inalámbricas. Amenazas en *Grid* y *Clusters* (temas de arquitectura, temas de infraestructura, temas de gestión,

temas de confianza). Sistemas de datos (amenazas a redes de áreas de almacenamiento, amenazas en sistemas de archivos distribuidos). Las amenazas a nivel de infraestructura se pueden analizar además desde el punto de vista de impacto y periodo de tiempo. Una amenaza particular aquí es el ataque de denegación de servicio considerado una amenaza inmediata y de impacto muy alto. Asegurar la infraestructura de red y los componentes (enrutadores, servidores, dispositivos inalámbricos) es una necesidad con carácter de urgencia.

- Amenazas a nivel de aplicaciones: En los últimos años ha habido un cambio tangible en el objetivo de los ataques pasando de los ataques a nivel de equipos y la red, a los ataques a nivel de las aplicaciones. Mientras las vulnerabilidades críticas son aplicables a todo tipo de aplicaciones, algunas vulnerabilidades son muy específicas a las aplicaciones Web. Una vulnerabilidad en una aplicación es fundamentalmente producida tanto por una deficiencia en el diseño o la aplicación de malas prácticas de programación. Estas últimas vulnerabilidades son introducidas dentro del código por una variedad de factores, pero la escases de conocimiento en seguridad por parte de los desarrolladores es una de las principales causas. Las principales vulnerabilidades y amenazas incluyen: gestión inapropiada de las sesiones, manejo inapropiado de los errores, deficiencia en la aplicación de técnicas de criptografía, errores de configuraciones; inyecciones, denegación de servicio, desbordamiento de memoria.

- Amenazas a nivel de servicios: Muchas de las vulnerabilidades en la capa de aplicaciones son en realidad variantes o mutaciones de los ataques encontrados en las capas de aplicaciones e infraestructura. Las vulnerabilidades en esta capa son provocadas por deficiencias en el diseño de los servicios o deficiencias en el código, provocado por malas prácticas de programación. Los mismos factores que contribuyen a la introducción de vulnerabilidades en la capa de aplicaciones, son aplicados en la capa de servicio. Las amenazas en la capa de servicio incluyen: fallos conocidas (*bugs)*, ataques de inyección SQL, inyecciones XPath y XQuery, inyecciones XPath a ciegas, sondeo al archivo WSDL (*WSDL proving*), ataques basados en parámetros, ataques de autenticación, ataques con intermediario (*man-in-the middle attacks*), ataques en el enrutamiento de los mensajes SOAP, virus adjuntados a los mensajes SOAP, ataques XML, ataques con análisis sintáctico complejos (*coercive parsing attack*), ataques basados en la manipulación de esquema, ataques al directorio UDDI.

2.1.2 Ataques DoS a Nivel de la Capa de Aplicación

Tal vez la más letal de las amenazas a nivel de los servicios, es la de denegar los servicios a los clientes. Este tipo de ataque es popularmente conocido como ataque de denegación de servicio (DoS)(Carl*, et al.*, 2006). En este tipo de ataque, la infraestructura de servicio es abrumada por una cantidad enorme de paquetes de datos que resultan en la denegación de los servicios a los usuarios o clientes de los servicios. Este ataque ha tenido un rápido crecimiento en las aplicaciones que se ejecutan sobre Internet siendo recientemente objeto de este ataque sitios Web conocidos (*Twitter* y *Facebook*) (Lemos, 2009). Este crecimiento en el número de ataques ha conducido a que los estudios de detección y prevención sean una de las áreas más activas de investigación en el campo de la seguridad.

Los ataques DoS pueden ser muy fáciles de generar pero muy difíciles de detectar, lo que les hace una arma muy atractiva para los atacantes. Este ataque puede ser agrupado en dos grandes categorías: ataques ordinarios y ataques distribuidos.

- Ataque DoS ordinario: Seleccionada una víctima en particular, el ataque DoS se genera mediante la saturación de los puertos con flujo de información, haciendo que el servidor se sobrecargue y no pueda seguir prestando servicios. Generalmente, el atacante utiliza una herramienta para enviar paquetes hasta lograr abrumar o deshabilitar a la víctima. Frecuentemente la dirección fuente de estos paquetes es falsificada haciendo difícil localizar la fuente real del ataque.

- Ataque DoS Distribuido (DDoS, *Distributed Denial of Service*): El ataque puede ser originado por un único atacante, pero el efecto del ataque se multiplica por el uso de servidores de ataques conocidos como "agentes". Al aunar los recursos de todos los sistemas comprometidos, se consigue un flujo que es capaz de saturar objetivos de gran potencia y gran ancho de banda. Podemos distinguir dos tipos de ataques por la estructura usada:

    - Ataques pre-programados: Puede utilizar un tipo de virus que incluye un ataque de denegación de servicio pre-programado. Al cumplirse una condición de tiempo, todos los ordenadores infectados realizan un ataque contra un objetivo seleccionado.

    - Ataques por control remoto: Este tipo de ataques utilizan máquinas comprometidas con un troyano para realizar ataques de denegación de servicio contra cualquier objetivo y en cualquier momento.

En el caso que nos interesa, los ataques de DoS a nivel de la capa de aplicación y servicio explotan específicamente vulnerabilidades en las aplicaciones. Este tipo de ataque tiene ventajas comparadas a los ataques DoS

tradicionales. Por ejemplo, las técnicas de monitorización de ataques DoS tradicionales son incapaces de detectar ataques de esta naturaleza ya que la mayoría de la veces los ataques son indistinguibles del tráfico normal. Además, la mayoría de estos ataques son muy difíciles de rastrear ya que la mayoría usan HTTP (*HyperText Transfer Protocol*) o HTTPs (*Hypertext Transfer Protocol Secure*) para llevar a cabo el ataque.

Algunos de los tipos de ataques DoS más frecuente a nivel de la capa de aplicación incluyen:

- Ataques DoS basados en XML (XDoS, *XML Based DoS Attacks*): En este tipo de ataque, los atacantes toman ventaja del proceso costoso de analizar documentos XML, conduciendo a un agotamiento de los recursos de la victima (memoria, CPU).

- Envenenamiento de Esquema (*Schema Poisoning*): Manipulando el esquema XML con el que se deben confrontar los mensajes, puede provocar un ataque DoS.

- Adulteración de direccionamiento (*Routing Detours*): La especificación *WS-Routing* provee los mecanismos para encaminar los mensajes hasta el destino a través de entornos complejos. Un ataque puede comprometer un nodo y causar un bucle dividiendo el entorno.

- Ataques de inyección SQL (*SQL injection attacks*): Permite lanzar diferentes tipos de ataques, ejecutando múltiples comandos desde un punto de entrada. Este ataque puede ser usado para manipular los datos e igualmente lanzar un ataque de desbordamiento de memoria, etc.

- Ataques de repetición (*Replay attacks*): Consiste en la repetición de mensajes antiguos aprovechando que los mensajes vienen de direcciones IP válidas.

Los ataques DoS a nivel de la capa de aplicación pueden ser agrupados en clases:

- Abuso de recurso y agotamiento: Consiste en exceder el uso esperado más allá de las capacidades del entorno.

- Explotación: Son tipos de ataques más sofisticados que analizan la aplicación y las funcionalidades e intentan identificar componentes vulnerables. Estos ataques pueden involucrar desbordamientos de memorias, o destrucción de la base de datos mediante ataques SQL letales.

Finalmente, las soluciones propuestas para detectar ataques DoS en la literatura pueden ser categorizadas ampliamente en dos grandes grupos: soluciones preventivas y soluciones reactivas. Las soluciones preventivas toman ventajas aplicando pasos preventivos, contrario a las soluciones reactivas que

apunta a identificar la fuente de los ataques. Algunas de las propuestas incluyen filtrado de paquetes, filtros a nivel de aplicaciones, sistemas de detección de intrusiones, pruebas de enlaces, entre otras. La siguiente sección se centra en el problema de la detección de intrusos.

2.2 Sistemas de Detección de Intrusión (IDS)

Los Sistemas de Detección de Intrusión (IDS, *Intrusion Detection Systems*), de ahora en adelante simplemente IDS, son sistemas que monitorizan el tráfico de una red y los sistemas de una organización en busca de señales de intrusión, actividades de usuarios no autorizados y la ocurrencia de malas prácticas, como en el caso de los usuarios autorizados que intentan sobrepasar sus límites de restricción de acceso a la información (Wu y Banzhaf, 2010). Además, un IDS dispone de los medios para manejar alertas cuando se detectan signos de intrusión que permitan tomar las medidas correspondientes en el menor tiempo posible.

Una intrusión se puede definir como toda actividad no autorizada que no debería ocurrir en el sistema. Una intrusión materializada (ataque) viola las políticas de seguridad de un sistema y compromete la integridad, confidencialidad y disponibilidad de los recursos.

Los IDS generalmente deben cumplir algunas características deseables independientemente de qué sistema vigile o su forma de trabajar, que incluyen:

- Una ejecución continúa de forma transparente, y con un mínimo de supervisión.
- Capacidad de tolerancia a fallos. Capacidad para recuperarse de posibles fallos o problemas con la red.
- Capacidad para identificar si alguno de sus componentes ha sido comprometido, e intentar recuperar dicho componente, y en caso contrario administrar un tipo de alerta.
- Minimizar el consumo de recursos.
- El IDS debe permitir aplicar una configuración según las políticas de seguridad que dicte la organización.
- Capacidad de adaptación. La capacidad de adaptarse a los rápidos cambios que sufren los sistemas y los usuarios. Además de incluir un proceso rápido y sencillo de actualización.

2.2.1 Clasificación de los IDS

Las primeras investigaciones sobre detección de intrusos se remontan al año 1980, cuando Jame P. Anderson (Anderson, 1980) planteó la posibilidad de mejorar la complejidad de auditoría y la habilidad para la vigilancia de sistemas informáticos. Es el primero en introducir el término *amenaza* dentro del campo de la seguridad Informática definiéndola como la posibilidad potencial de un intento deliberado de acceso y manipulación de la información, o lograr que un sistema quede inutilizable.

Entre 1988 y 1990 el Instituto de Investigación *SRI International* desarrolla IDES (*Intrusion Detection Expert System*), un sistema experto que detecta las desviaciones a partir del comportamiento de diferentes sujetos (Lunt, 1990), (Lunt y Jagannathan, 1988), siendo el primer sistema de detección de anomalías en equipo (*host)*. Mientras tanto, en los laboratorios *Lawrence Livermore* de la Universidad de California en Davis, se desarrolla el proyecto *Haystack,* el cual se convertiría en el primer IDS de usos indebidos basado en firmas. *Haystack* analizaba los datos de auditoría y los comparaba con patrones de ataque predefinidos (Smaha, 1988). Seguidamente empiezan a aparecer los primeros proyectos de IDS basados en red tales como NSM (*Network Security Monitor*) (Heberlein*, et al.*, 1990), NADIR (*Network Anomaly Detector and Intrusion Reporter*) (Jackson*, et al.*, 1990).

Estas primeras investigaciones de los IDS sentaron la base para posteriormente proponer nuevos proyectos. Actualmente, los IDS hacen uso de diferentes técnicas y algoritmos para analizar el comportamiento de los sistemas informáticos.

Existen varios tipos de IDS (Wu y Banzhaf, 2010), clasificados según la fuente de datos, el tipo de detección que posee y la estrategia de respuesta. En la Figura 2.3 se representa la clasificación de lo IDS.

- Fuentes de información. Se refiere a la fuente u origen de los datos que se utilizan en el análisis para determinar si una intrusión se ha llevado a cabo.

- Análisis: Se refiere al método de detección utilizado como estrategia. Diferentes técnicas y algoritmos pueden ser aplicados como métodos de detección de intrusión.

- Respuestas. Una vez se ha determinado si ha sucedido alguna intrusión, los IDS pueden o bien responder de forma activa ante la misma, o bien registrar la detección y no realizar acción alguna.

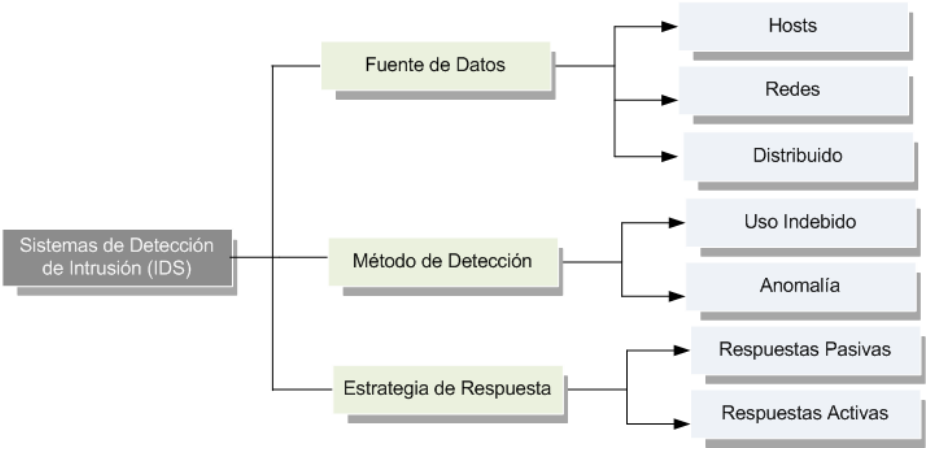

Figura 2.3. Clasificación de los sistemas de detección de intrusión

2.2.1.1 Basado en la Fuente de Información

Generalmente los IDS requieren manejar una multitud de datos provenientes de distintas fuentes para intentar identificar la presencia de algún tipo de intrusión. Los datos recolectados se pueden diferenciar en dos grupos; los datos obtenidos de una máquina o equipo (*host*), y los datos obtenidos de una red a través de una monitorización. A partir de cada grupo, se pueden identificar el tipo de enfoque que se pueden tomar:

- Detección de Intrusión basado en equipo (HIDS, *Host Intrusion Detection System*): Analizan el tráfico sobre un servidor o una computadora; están dedicados a monitorizar lo que está sucediendo en cada equipo y son capaces de detectar situaciones como los intentos fallidos de acceso o modificaciones en archivos críticos.

- Detección de Intrusión basada en Red (NIDS, *Network Intrusion Detection System*): Analizan el tráfico de la red completa, examinando los paquetes individualmente e identificando comportamientos fuera del perfil normal no detectados por los cortafuegos.

- Detección de Intrusión Distribuida: (DIDS, *Distributed Intrusion Detection System*): Basado en la arquitectura cliente-servidor compuesto por una serie de NIDS (IDS de redes) que actúan como sensores centralizando la

información de posibles ataques en una unidad central que puede almacenar o recuperar los datos de una base de datos centralizada.

2.2.1.2 Basado en el Método de Detección

Los dos tipos más conocidos de detecciones que un IDS puede realizar son:

- Detección de uso indebido: Compara patrones (*signatures*) de ataque conocidos previamente en busca de coincidencias. La principal desventaja de este tipo de detección es que no se puede detectar nada que no haya sido establecido al definir el conjunto de firmas. Como consecuencia, los IDS basados en este tipo de detección son incapaces de detectar nuevos tipos de ataques, teniendo que actualizar continuamente su base de datos de firmas.

- Detección basada en anomalía: Los IDS basados en anomalía se basan en la premisa de que cualquier ataque o intento implica un uso anormal de los sistemas.

En este sentido hay dos grandes aproximaciones; la primera se refiere a la capacidad del IDS para aprender por sí mismo, tomando en cuenta por ejemplo el comportamiento de los usuarios, de sus procesos, del tráfico de la red, etc. La primera aproximación utiliza básicamente métodos estadísticos, aunque en los últimos años la aplicación de estrategias de Inteligencia Computacional (*Computacional Intelligence* o *Soft Computing*) se ha convertido en uno de los campos de investigación más prominente dentro de los IDS (Wu y Banzhaf, 2010), (Kandeeban y Rajesh, 2010). Este nuevo paradigma se ha venido desarrollando para solucionar diferentes clases de problemas que no han podido ser resueltos a través de los métodos tradicionales (Sadkhan, 2009). Las nuevas estrategias de detección en los IDS incluyen redes neuronales artificiales, conjuntos de lógica difusa, computación evolutiva, sistemas inmunológicos artificiales y sistemas de colonias. Estos enfoques, excepto por los conjuntos de lógica difusa, son capaces de adquirir e integrar autónomamente conocimiento y pueden ser implementados ya sea a través de un modo de aprendizaje supervisado o no supervisado.

La segunda aproximación especifica que el sistema requiere la introducción de dicho comportamiento mediante un conjunto de reglas, basándose en determinados parámetros de los sistemas, pero afrontando el problema a la hora de decidir con precisión cuáles parámetros delimitan los comportamientos intrusivos. El enfoque basado en especificaciones fue inicialmente propuesto y desarrollado por investigadores de la Universidad de California en Davis (Ko*, et al.*, 1997).

A manera de resumen se presenta un esquema, Figure 2.4, de las técnicas de detección desde el punto de vista de la estrategia de análisis basada en las prospecciones realizadas en (Lazarevic*, et al.*, 2005) y (Noel*, et al.*, 2002)

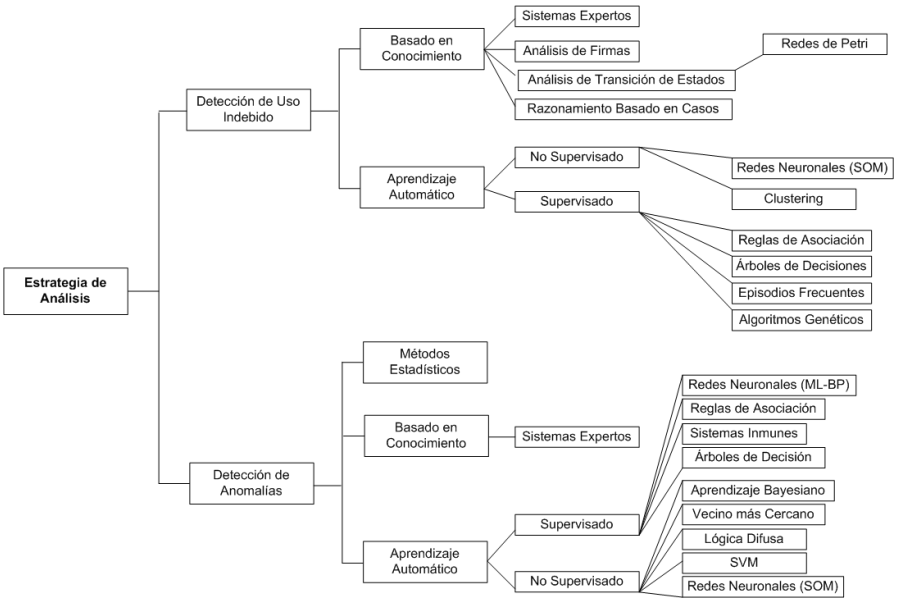

Figure 2.4. Resumen de las técnicas de detección de intrusión

2.2.1.3 Basado en la Estrategia de Respuesta

Esta clasificación considera la reacción del IDS frente a un posible ataque:

- Pasivos: Los IDS con esta estrategia de repuesta registran la información y generan una notificación que no es más que un mensaje enviado a través de una interface habilitada en el mismo IDS, un correo electrónico, SMS, etc.

- Activos: Están diseñados para responder ante una actividad ilegal. Este es un campo en investigación debido a la complejidad de automatizar una respuesta. Es necesario considerar el coste de una intrusión para estimar la amplitud de la respuesta para evitar agravar el problema de la intrusión.

2.2.2 Ventajas y Desventajas de los IDS

Evidentemente, los IDS, como cualquiera otro sistema de esta naturaleza, presentan ventajas y desventajas. La habilidad está en saber combinar las características positivas para lograr una mayor efectividad. En la Tabla 2.1 se resumen las ventajas y desventajas más importantes de los IDS atendiendo a la fuente de datos y la estrategia de análisis.

Tabla 2.1. Ventaja y desventaja de los IDS

Perspectiva	Ventaja	Desventaja
Ubicación	Host	Host
	• Monitorizan actividades más específicas del sistema (acceso a ficheros, intentos de instalación, etc.) • No requieren ningún tipo de hardware, basta con un paquete de software en el equipo. • Presentan un buen desempeño en redes de alta velocidad. • Dependiendo de la localización, pueden revisar información cifrada	• Dependen del Sistema operativo del host. • Requieren una instalación en cada host donde se va a monitorizar • Para seguir el rastro de la fuente de ataque requiere información de un ataque previo. • Pueden ser deshabilitados por ciertos tipos de ataque.
	Red	Red
	• Son independiente del sistema operativo. • Pueden reducir el costo al cubrir una red entera o un segmento considerado vulnerable. • Pueden detectar los ataques antes que alcance el objetivo al examinar el tráfico en busca de actividades sospechosas. • Son más eficientes a la hora de seguir el rastro a la fuente de ataque. • Son capaces de detectar intentos de intrusiones fallidos.	• Dependen de la topología y protocolo de red. • La monitorización de actividades de los sistemas es menos detallada. • Requieren un tipo de hardware para su instalación. • Pueden fallar en la detección en periodos de tráfico alto. • No pueden analizar tráfico cifrado.
Análisis	Uso Indebido	Uso Indebido
	• Son muy efectivos en la detección de ataques sin que generen un número elevado de falsas alarmas. • Pueden rápidamente y de forma precisa diagnosticar el uso de una herramienta o técnica de ataque.	• Sólo son capaces de detectar lo que conocen. Nuevos ataques no son reconocidos. • Requiere una actualización continua de su base de dato de firmas. • Incapacidad para detectar ataques camuflados.
	Anomalía	Anomalía

	• Construyen perfiles representando el comportamiento normal de los usuarios de forma autónoma sin necesidad de actualizar firmas continuamente. • Pueden detectar comportamientos inusuales identificando ataques para los cuales no tienen un conocimiento específico. • Pueden generar información que puede ser utilizada para definir firmas en los IDS de uso indebido.	• Genera alta tasa de falsos positivos • Requiere mucho tiempo para desarrollar un modelo preciso. Requieren grandes conjuntos de entrenamiento. • Pueden ser engañados por expertos furtivos conocedores de la red.

Otro punto importante a considerar es la zona de localización del IDS en la red. Para ello se han establecido tres zonas:

- Zona roja: Es la zona considerada de alto riesgo. Con una configuración estricta puede generar muchas alarmas porque puede ver todo el tráfico que entra y sale de nuestra red.

- Zona verde: Requiere una configuración un poco más sensible ya que el cortafuego ha filtrado gran parte del tráfico. No debería generarse tantas falsas alarmas ya que el acceso estaría permitido solo a los servidores bajo protección.

- Zona azul: Considerada la zona de confianza; cualquier tráfico anómalo que llegue a este punto debe ser considerado hostil. Es el lugar donde se producirá el menor número de falsas alarmas, requiriendo atención inmediata.

Finalmente es importante resaltar que los IDS no pueden automatizar completamente la investigación de los incidentes, lo que obliga a la intervención de un experto para descubrir la naturaleza real de un ataque, limpiar sus efectos y tomar las medidas necesarias para protegerse para el futuro.

2.2.3 Arquitecturas IDS Distribuidas Basadas en Agentes

La demanda actual de efectividad en los IDS y la complejidad de los nuevos entornos ha generado el desarrollo de nuevos paradigmas en el campo de los IDS. Uno de estos nuevos paradigmas son los sistemas basados en agentes y sistemas multi-agente. Los agentes poseen características que los hacen idóneos para abordar problemas en la detección de intrusión. Por ejemplo, los agentes son capaces de actuar autónomamente en un entorno, se pueden comunicar directamente con otros agentes, se pueden adaptar para tratar nuevas amenazas y reducen los costes de comunicación (Abraham*, et al.*, 2007). No obstante, las tareas de un sistema IDS, requiere más que un único agente. En ese sentido, se plantea la implementación de varios agentes cooperando entre sí para realizar todas las tareas de un determinado problema. Este grupo de agentes trabajando en conjunto forma una estructura conocida como sistema multi-agente, donde cada agente coopera para solucionar problemas que van más allá de las capacidades y el conocimiento individualizado que posee cada agente.

La incorporación de Sistemas Multi-agente dentro del campo de los IDS se ha convertido en un área de investigación importante en los últimos años (Herrero y Corchado, 2009). Los sistemas multi-agente se centran en resolver tareas de detección a nivel de red o en entornos distribuidos. Además, los agentes ofrecen la capacidad de ser combinados con otras técnicas de la Inteligencia Artificial para dotarlos de mayores capacidades de aprendizaje y autonomía.

A continuación presentamos una revisión de las aplicaciones de sistemas multi-agente aplicados a la detección de intrusión, tomando en cuenta las revisiones llevadas a cabo por (Abraham*, et al.*, 2007), (Herrero y Corchado, 2009).

- MOVIH-IDS (*Mobile-Visualization Hybrid IDS*) (Herrero*, et al.*, 2009): Construido sobre la base de un sistema multi-agente incorporando una red neuronal artificial para la visualización del tráfico de red. Incluye agentes deliberativos usando razonamiento basado en caso. Implementa un modelo conexionista no supervisado para identificar intrusiones en el tráfico de las redes.

- PAID (*Probabilistic Agent-Based IDS*) (Gowadia*, et al.*, 2005): Es una arquitectura de agentes cooperativos donde los agentes autónomos ejecutan tareas específicas de detección de intrusión. Maneja 3 tipos de agentes para tares de monitorización, análisis y coordinación. La principal novedad dentro de la arquitectura es el modelo que permite a los agentes compartir sus creencias (*beliefs*).

- CIDS (*Cougaar-based IDS*) (Dasgupta*, et al.*, 2005): Presenta una Infraestructura de un sistema distribuido para la detección de intrusión que

emplea 4 tipos de agentes para ejecutar funciones de administración, toma de decisiones, respuesta y monitoreo. Fue desarrollado sobre la infraestructura *Cougaar* (BBN Technologies, 2009). Utiliza módulos para el soporte a la toma de decisiones de una forma inteligente. El sistema emplea una base de datos de conocimiento de ataques conocidos y un motor de inferencia difusa para clasificar actividades de la red (legales o maliciosas).

- SPIDeR-MAN (*Synergistic and Perceptual Intrusion Detection with Reinforcement in a Multi-Agent Neural Network*) (Miller*, et al.*, 2003). Cada agente usa un SOM (*Self-Organizing Map*). Un clasificador basado en reglas para detectar actividades maliciosas. Un mecanismo tipo pizarra se utiliza para recoger los resultados los generados por los agentes (decisión en grupo). Se utiliza el aprendizaje reforzado a través de una señal de refuerzo (recompensa/castigo) generada dentro de la pizarra y distribuida a todos los agentes participantes en la toma de decisión.

- Un sistema detección de intrusión basado en agentes para la detección de tres tipos de amenazas (Denegación de servicio, robo de documentos codificados y el *ping sweep*) fue presentado por (Hegazy*, et al.*, 2003). El sistema está basado en 4 módulos: un modulo de *sniffing*, modulo de análisis, modulo de decisión y un modulo de reporte. Un conjunto de agentes *sniffer* no especializados escuchan en la red extrayendo los paquetes requeridos. El análisis y la toma de decisión es ejecutado por tipos de agentes especializados. Finalmente para el modulo de reporte utiliza un agente simple para generar reportes o hacer los registros. Dependiendo de la función de cada agente, se dota a los agentes de una estructura interna más especializada.

- Una arquitectura multi-agente para la detección de intrusiones en entornos de redes locales fue propuesto por (Zhang*, et al.*, 2001). La arquitectura incluye 4 tipos de agentes para ejecutar funciones de monitorización (en equipos, segmentos de red y servidores públicos); coordinación e interacción con el administrador mediante interface. La arquitectura está basada en una estructura jerárquica.

- AAFID (*Autonomous Agents For Intrusion Detection*) (Spafford y Zamboni, 2000): Es un enfoque IDS distribuido donde las entidades están organizadas dentro de una arquitectura jerárquica con control centralizado. Emplea agentes autónomos, considerando a los agentes como agentes de software para ejecutar funciones de monitorización a nivel de los equipos. Está compuesta por un conjunto de compontes siendo los agentes los encargados de monitorizar ciertos aspectos en los diferentes equipos bajo observación. En los detalles de la arquitectura no se indica la estructura interna de los agentes. Los agentes en AAFID no se comunican directamente entre ellos.

- *Open Infrastructure for Scalable Intrusion Detection* (Reilly y Stillman, 1998): Es una infraestructura para la detección de intrusión basada en agentes, construida en base a *Common Intrusion Detection Framework* (CIDF) (Kahn, *et al.*, 1998), este último consiste en un infraestructura general para el desarrollo de sistemas de detección de intrusión. La infraestructura define una jerarquía de capas con diferentes tipos de agentes para las tareas requeridas (respuesta a las intrusiones, recolección de información, análisis de información y administración de la propia arquitectura IDS).

- The JAM Project (*Java Agents for Metalearning*) (Stolfo, *et al.*, 1997): Combina agentes inteligentes y técnicas de minería de datos. Utilizando un algoritmo basado en reglas de asociación se determina la relación entre los diferentes campos de un registro de auditoría, mientras un clasificador de meta-aprendizaje aprende las firmas de ataques. En base a las características extraídas por las técnicas de minería de datos se construye modelos de comportamientos maliciosos.

Cabe mencionar dentro de las arquitecturas IDS distribuidas basadas en agentes, los enfoques basados en agentes móviles. En estos enfoques los agentes viajan entre diferentes equipos conectados a la red para monitorizarlos. Entre estas propuestas podemos mencionar IDA (ID Agent) (Asaka, *et al.*, 1999) apuntado a detectar varias intrusiones eficientemente en vez de detectar todo el conjunto de intrusiones con precisión. En vez de monitorizar las actividades de los usuarios, la estrategia consiste en detectar los eventos que pueden relacionarse con intrusiones. IDA recoge información adicional, analiza y decide si una intrusión ha tenido lugar. SPARTA (*Security Policy Adaptation Reinforced Through Agents*) (Krüegel y Vigna, 2003) es propuesto como una arquitectura para recoger y relacionar datos de detección de intrusión distribuidos usando agentes móviles. Los agentes móviles en SPARTA habilitan el análisis distribuido, mejoran la escalabilidad e incrementan la tolerancia a fallos. Los agentes móviles están encargados de recoger la información distribuida para responder las solicitudes de usuarios. MAIDS (*multi-agent IDS*) (Helmer, *et al.*, 2002) es una arquitectura incorporando agentes móviles ligeros, localizados en la mitad de la arquitectura para formar la primera línea de detección de intrusión. Los agentes viajan periódicamente entre sistemas monitorizados obteniendo información y analizando los datos para determinar si una intrusión específica ha ocurrido. IDReAM (*Intrusion Detection and Response executed with Agent Mobility*) (Foukia, 2005) es un enfoque para construir un sistema de respuesta y un identificador de intrusión completamente distribuido y descentralizado. Conceptualmente IDReAM combina agentes móviles con paradigmas de auto-organización inspirado por sistemas naturales como los sistemas inmunológicos. IDReAM es evaluado en términos de consumo de recursos y eficiencia en la respuesta de intrusiones. Finalmente, una infraestructura basada en agentes móviles fue propuesta por (Wang, *et al.*, 2006). Algunos de los componentes son diseñados como agentes móviles buscando una adaptabilidad alta y seguridad del sistema.

Se argumenta que estos agentes móviles pueden evadir intrusiones y recuperarse por sí mismos después de ser comprometidos.

Los sistemas de detección distribuidos se presentan como una solución sólida a muchos de los problemas de seguridad encontrados en sistemas de redes y entornos complejos; no obstante, existen muchos desafíos y temas por resolver en este campo (Abraham*, et al.*, 2007). Algunos de los desafíos incluyen cómo manejar actividades de ataque que están ocurriendo en múltiples localizaciones cuando cada sistema de detección tiene acceso directo a un punto único de información. Otro importante desafío está relacionado con la compartición de información distribuida, que es el componente clave para un sistema de detección distribuido robusto. En la sección 2.3 se presentan los ataques de inyección SQL y XML, en los que se centra este trabajo de tesis doctoral.

2.3 Ataques de Inyección SQL y Ataques XDoS en la Capa de Aplicación

En esta sección se presentan dos de los principales ataques IDS: Los ataques de inyección SQL y los ataques XDoS. Se trata de los ataques que presentan una mayor problemática en la actualidad y que requieren de soluciones innovadoras.

2.3.1 Ataques de Inyección SQL

Las aplicaciones para trabajar en Internet han experimentado un rápido crecimiento en los últimos años (Jazayeri, 2007). La mayoría de estas aplicaciones hacen uso de algún producto de base de datos para gestionar y proporcionar información actualizada, y ofrecer nuevos servicios a sus usuarios. La gestión de la información en las bases de datos se realiza a través del lenguaje SQL (*Structure Query Language*) (Limeback, 2008), el cual constituye la columna vertebral de la mayoría de los sistemas de bases de datos modernos, especialmente de la bases de datos relacionales (Patrick, 2009). Sin embargo, una amenaza potencial a nivel de las aplicaciones desarrolladas para trabajar en Internet ha puesto en riesgo la confidencialidad, integridad y disponibilidad de la información almacenada en las base de datos. Nos referimos al ataque de inyección de código, conocido como ataque de inyección SQL (Anley, 2002), (Litchfield*, et al.*, 2005), (Halfond*, et al.*, 2006). Esta amenaza a nivel de la capa de aplicación no es reciente, sino que una serie de trabajos (Rain Forest, 1998), (Allaire Security Bulletin, 1999), (Astley y Puppy, 1999) fueron publicados años

atrás demostrando como los productos disponibles para el desarrollo de aplicaciones eran vulnerables a código SQL. Sin embargo, el término "inyección" como tal no fue acuñado de inmediato; después de varias publicaciones sobre demostraciones del ataque fue cuando se empezó a generalizar el término "inyección SQL"(Litchfield, 2005).

Los ataques de inyección SQL pueden causar serios daños. Los problemas de seguridad más habituales implican evasión de los mecanismos de autenticación, modificación de los datos sin autorización, descarga de archivos desde el servidor de base de datos comprometido, carga de archivos con código malicioso, ejecución de comandos remotos en el servidor, interrupción de los servicios o del sistema, escalamiento de privilegios y robo de información confidencial. Los últimos informes de estadísticas en relación a los ataques de inyección SQL dejan ver claramente que es uno de los ataques más frecuente en los últimos años contra las aplicaciones en Internet. La Figura 2.5 extraída del reporte de seguridad de IBM (IBM Corporation, 2009) en el año 2008 muestra como el ataque de inyección SQL ha ido en constante ascenso a partir del año 2007.

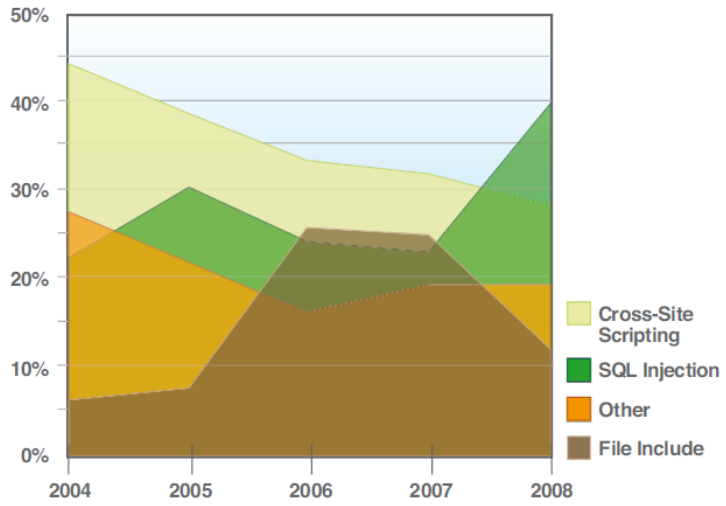

Figura 2.5. Evolución del ataque de inyección SQL en los últimos 5 años

El lenguaje SQL incluye diferentes tipos de comandos. Los comandos usados para definir, modificar y eliminar la estructura de la base de datos (*Data Definition Language o DDL*). Los comandos para manejar los datos almacenados en la base de datos (*Data Manipulation Language Statment o DML*) y los comandos de control para mantener la seguridad de los datos almacenados (*Data Control Language o DCL*). La disponibilidad de un determinado comando viene determinada por la implementación SQL que incorpora la base de datos.

Los ataques de inyección SQL utilizan las vulnerabilidades en las aplicaciones y aprovechan funcionalidades extras en los sistemas de base de datos. Un ataque de inyección SQL se produce cuando la consulta original es modificada mediante la inyección de elementos propios del lenguaje SQL de forma arbitraria y no autorizada. El medio tradicional para ejecutar un ataque de inyección SQL es a través de las interfaces de las aplicaciones, donde los usuarios proporcionan los parámetros de entrada. Generalmente, los ataques de inyecciones SQL aprovechan la ausencia de mecanismos de validación sobre los parámetros de entrada, los cuales son concatenados de forma directa a la cadena de texto de la consulta SQL. Una vez construida la consulta SQL, ésta es enviada a la base de datos para su procesamiento, materializándose el ataque.

Los ataques de inyección SQL han ido evolucionando, dando origen a varias tipos de ataque, como se hace referencia en (Halfond*, et al.*, 2006) y (D. B. Networks, 2009). En la Tabla 2.2 se presentan las tipos más comunes de ataques de inyección SQL.

Tabla 2.2. Tipos de ataques de inyección SQL

Tipo de inyección	Descripción	Ejemplo
Tautologías	Se inyecta código en la cadena SQL, de la consulta, en una o en todas las declaraciones condicionales, de forma que siempre sean evaluadas como verdaderas. En este caso se aprovecha un parámetro de entrada vulnerable, usado para construir la condición. El ataque resulta efectivo, cuando de la tabla objetivo, se logra recuperar cuando menos un registro.	SELECT * from TbllUser where Username = '' or 1=1--and pass= ''
Errores Lógicos / Consultas Ilegales	Se aplica reingeniería sobre los mensajes de errores, enviados desde la base de datos como respuesta, para obtener información acerca del esquema de la base de datos. Un ejemplo es la adición de la comilla dentro de la cadena de texto suministrada desde el parámetro de entrada.	SELECT * FROM TblSupplier WHERE NameSupplier = 'O'Reill'
Basados en e l Operador *Union*	Es posible modificar el conjunto de resultados de la consulta original, cambiando su lógica. A través de un parámetro de entrada vulnerable, se adiciona una segunda consulta mediante el operador *union.* Es posible recuperar resultados de diferentes tablas producto de la unión de las tablas de la consulta original y de la segunda consulta adicionada.	SELECT * FROM TblSupplier WHERE NameSupplier = '' UNION ALL SELECT * From TblConsumer WHERE 1 = 1

Piggy-backed Queries	Este tipo de ataque no modifica la lógica de la consulta original. A su vez, adiciona una nueva consulta totalmente diferente a la primera, con lo que la base de datos recibe más de una consulta SQL. Este ataque depende de la configuración de la base de datos, cuando permite múltiples consultas en una única cadena de texto SQL.	SELECT * FROM TblUser WHERE UserName = '' ; DROP TABLE TblUser—'AND pass = ''
Inyección basada en Inferencia	La inyección *a ciegas* permite inferir tomando en cuenta el comportamiento de la página cuando se envía una pregunta (cierto/falso) al servidor. Si la respuesta es verdadera, la aplicación continúa con su funcionamiento normal. En caso contrario, el funcionamiento de la página difiere del habitual. El *timing attacks* es similar a la inyección *a ciegas*, pero utiliza un método de inferencia diferente. La estructura de la consulta a ejecutar es un enunciado de la forma *SELECT IF(expression, true, false)* donde en unas de las ramas se incluye una función de tiempo. Por ejemplo *BENCHMARK()*o *WAITFOR DELAY*. Midiendo el incremento o decremento del tiempo de repuesta, es posible determinar la rama seguida por la consulta e inferior en la respuesta.	Declare @s varchar(8000); select @s = db_name(); if (ascii(substring(@s, 1, 1)) & (power(2, 0))) > 0 waitfor delay '0:0:5'
Inyección basada en Stores Procedures	Lo primero es conocer cuál producto de base de datos está dando soporte. El problema se deriva de la extensión en la funcionalidad que la mayoría de los productos de bases de datos ofrecen mediante la incorporación de procedimientos almacenados. Estos procedimiento almacenados pueden incluso interactuar con el sistema operativo. Es posible elaborar consultas maliciosas que aprovechen la funcionalidad de un procedimiento almacenado para tomar el control del servidor o bloquearlo.	Simplequoted.asp?city=seattle';EXEC master.dbo.xp_cmdshell 'cmd.exe dir c:
Codificación Variable	Más que un tipo de ataque, esto permite evadir los mecanismos de detección y prevención de intrusiones. Mediante la codificación, es posible ocultar a los mecanismos de seguridad patrones conocidos de ataque. Para codificar las cadenas de texto, se utilizan varios métodos de codificación tales como hexadecimal, ASCII o UNICODE). Esta técnica de codificación hace difícil plantear algún mecanismo de seguridad basado en la revisión de código, sencillamente porque resulta difícil manejar todas las posibles variaciones.	Ej.: declare @q varchar(8000); select @q = 0x73656c656374204040766 57273696f6e; exec(@q) El resultado de la codificación es 'select @@version'

Combinación de Ataques	Las técnicas descritas arribas se pueden combinar para alcanzar más de un objetivo a la vez. Por ejemplo, es posible recuperar información del esquema de la base de datos mediante consultas ilegales, al mismo tiempo que se apunta a identificar el producto de la base de datos para aprovechar el uso de procedimientos almacenados que den acceso al sistema operativo. Todo ello, oculto bajo un método de codificación que no sea detectado por los mecanismos de seguridad.	

Los ataques de inyección SQL se centran en las aplicaciones sobre Internet, con lo cual manejan información valiosa al estar conectada a una base de datos corporativa; otro factor importante es que las aplicaciones están disponibles las 24 horas del día, y son accesibles desde cualquier lugar. Finalmente, las soluciones de seguridad implementadas para proteger las aplicaciones y la base de datos similares a los cortafuegos (*firewall*) y sistemas de detección y prevención de intrusión (IDS/IPS) no detectan o detectan con poca precisión este tipo de amenaza. En el caso de los cortafuegos, se configuran por defecto dejando abiertos los puertos TCP 80 y 443 (HTTP y HTTPS) para permitir el tráfico web. Al dejar pasar el tráfico Web sin ninguna comprobación, los ataques de inyección SQL pasan ocultos hasta llegar a la base de datos que suministra los datos a las aplicaciones.

Los ataques de inyección SQL han sido estudiados desde diferentes enfoques. En (Halfond*, et al.*, 2006) se han revisado los diferentes enfoques existentes y se ha llevado a cabo una categorización de los enfoques propuestos. Nosotros hemos tomando en cuenta está organización de los enfoques y la hemos actualizado adicionando los enfoques recientes.

Como la raíz del ataque de inyección SQL se debe a una validación insuficiente en las entradas de las aplicaciones, la principal recomendación para reducir o eliminar el problema es la escritura de código seguro mediante la aplicación de una "programación defensiva" (*defensive coding practice*) durante la fase de desarrollo. Generalmente esto incluye revisión de los tipos de datos proporcionados como entradas a la aplicación; codificación de los meta-caracteres, para ser interpretados como caracteres normales por la base de datos; limitar la longitud de las entradas (tipo cadenas); controlar los mensajes de excepciones retornados por la aplicación o la base de datos en eventos de errores, e identificación y comprobación de todas las fuentes de entrada en la aplicación. No obstante, aunque la aplicación de una programación defensiva es la mejor forma de prevenir los ataques de inyección SQL, esto puede resultar en la mayoría de los casos complicado. Esto depende de la habilidad de los programadores para una rigurosa implementación, siendo propensa a errores humanos.

Del otro lado, existe un número considerable de enfoques que han sido propuestos para la prevención, detección y bloqueo de ataques de inyección SQL. Los enfoques en esta categoría han sido agrupados según el tipo de técnica utilizada para abordar los ataques de inyección SQL. En este punto haremos una revisión rápida de las diferentes técnicas estudiadas y nos centraremos en los enfoques basados en detección de intrusión, los cuales se relacionan con nuestro trabajo.

De las técnica propuesta para la prevención y detección de ataques están los enfoques basados en los test de penetración (Huang*, et al.*, 2003) y (Sekar, 2009) para detectar puntos vulnerables en la aplicación. Generalmente con estos enfoques se intenta identificar todos los puntos vulnerables de la aplicación o capturar las solicitudes enviadas para luego evaluar la respuesta de la aplicación con entradas pre-elaboradas. En la misma línea, está la técnica basada en el análisis sintáctico, implementada en diferentes enfoques(Christensen*, et al.*, 2003), (Gould*, et al.*, 2004), (Wassermann y Su, 2004), (Wassermann*, et al.*, 2007), (Fu*, et al.*, 2007), (Fu y Qian, 2008) para detectar puntos vulnerables en el código de la aplicación. Algunos de estos enfoques no han sido propuestos específicamente para detectar y bloquear ataques de inyecciones SQL pero pueden ayudar a prevenir los tipos más comunes de ataque.

Las técnicas de análisis sintáctico han sido combinadas con análisis en tiempo de ejecución para crear modelos y algoritmos más eficaces en la detección. Varios enfoques han sido propuestos (Halfond y Orso, 2005a), (Halfond y Orso, 2005b), (Buehrer*, et al.*, 2005), (Su y Wassermann, 2006), (Muthuprasanna*, et al.*, 2006), (Shin*, et al.*, 2006), (Kosuga*, et al.*, 2007), (Bandhakavi*, et al.*, 2007), (Kosuga*, et al.*, 2007), (Junjin, 2009) utilizando está estrategia. Generalmente se analiza el código de la aplicación para identificar puntos vulnerables y luego se generan modelos para compararlos con el modelo de la nueva petición en tiempo de ejecución y así determinar irregularidades.

Siguiendo la estrategia de ejecutar un tipo de análisis, la técnica de "chequeo" de contaminación "*taint checking*" ha sido planteada en ciertos enfoques (Huang*, et al.*, 2004), (Nguyen-Tuong*, et al.*, 2005), (Pietraszek y Berghe, 2005), (Haldar*, et al.*, 2005), (Martin*, et al.*, 2005). Mediante la aplicación de un análisis sensitivo al contexto se busca identificar y rechazar las consultas SQL maliciosas que se han generado a partir de entradas poco confiables e identificar la ubicación de estas entradas.

Adicional a las técnicas antes mencionadas, otros tipos de técnicas se han direccionados para hacer frente a los ataques de inyección SQL. Algunas de estas técnicas proponen nuevos paradigmas para el desarrollo de consultas como en (McClure y Krüger, 2005), (Cook y Rai, 2005), (Bravenboer*, et al.*, 2007), (Shahriar y Zulkernine, 2008), (Thomas y Williams, 2007). A nivel de componente sobre la capa de aplicación, similares a un proxy se ha propuesto *security Gateway* (Scott y Sharp, 2002), el cual filtra los ataques aplicando reglas de validación a las entradas. Otra estrategia planteada es el creación y uso de un

conjunto instrucciones aleatorias aplicadas a las sentencias claves del lenguaje SQL (Boyd y Keromytis, 2004), el cual permite crear consultas usando instrucciones aleatorias en vez de las sentencias SQL regulares.

Finalmente, están los enfoques cuya técnica está basada en la detección de intrusión. Sobre este conjunto de enfoques centraremos nuestra atención al guardar relación con nuestra propuesta en la mecánica aplicada para resolver el problema. No obstante, dentro de este conjunto existen propuestas que abordan directamente los ataques de inyección SQL y otros que los abordan indirectamente a través de su estrategia de detección. Estos últimos serán mencionados brevemente para centrarnos en los trabajos que direccionan concretamente este ataque.

Los enfoques basados en detección de intrusión y que cubren de alguna manera, dentro de su portafolio de amenazas las inyecciones SQL, emplean diferentes estrategias propias de los sistemas de detección de intrusión (IDS). Es el caso de (Krüegel y Vigna, 2003), (Bertino*, et al.*, 2004), (Chen*, et al.*, 2005) y (Cova*, et al.*, 2007). Estos enfoques, aunque no afrontan directamente los ataques de inyección SQL, su estrategia de detección puede detectar ciertos tipos de ataques. Las estrategias de detección incluyen análisis de los parámetros de entrada en las solicitudes, análisis del perfil de acceso a la base de datos, análisis del registro histórico de las consultas (*database log*), análisis y monitoreo de la aplicaciones con acceso a la base de datos, entre otros. Como una característica de los IDS, estos enfoques generalmente manejan una etapa de aprendizaje para crear una base de conocimiento y una etapa de detección para capturar y bloquear los ataques. Entre las principales desventajas que presentan estos enfoques se puede mencionar la sobrecarga impuesta a raíz de las complejas tareas del proceso de detección, y el problema de precisión en la detección (falsos positivos y falsos negativos).

Los ataques de inyección SQL se han convertido en una de la amenazas más frecuente a nivel de las aplicaciones en Internet en los últimos años. Es por ello que un número considerable de trabajos han sido propuestos como solución al problema; sin embargo los ataques de inyecciones SQL siguen siendo un problema sin resolver y una amenaza latente. Después de haber llevado a cabo una revisión amplia de las diversas técnicas y enfoques existentes como solución propuesta a este ataque en concreto, nos centraremos ahora en los trabajos basados en detección de intrusión, que tratan de forma específica este ataque en su estrategia de detección. Como nuestra propuesta plantea un mecanismo para la detección y bloqueo de inyecciones SQL desde el enfoque de un sistema de detección de intrusión, resulta provechoso llevar a cabo una comparación para presentar de forma clara donde nuestra propuesta mejora los enfoques existentes. A continuación revisaremos estos enfoques y en el capítulo de conclusiones presentaremos una tabla comparativa.

- Un enfoque basado en detección por anomalía fue propuesto por (Bockermann*, et al.*, 2009). El enfoque propone modelar la consulta SQL a través del árbol sintáctico de la cadena SQL. Considera la importancia del contexto (estructura de la cadena SQL) cuando se llevan a cabo un tipo de análisis sobre consultas SQL. Mediante una estrategia de agrupamiento (*clustering*) se agrupan las consultas similares y se aíslan las consultas consideradas maliciosas. Su principal desventaja está en su alta sobrecarga computacional, lo que afectaría una detección en tiempo real.

- Un enfoque combinando métodos basados en firmas (*Signature*) y métodos de auditoría ha sido recientemente propuesto por (Ezumalai y Aghila, 2009). Con la implementación del algoritmo *Hirschberg* se busca reducir el tiempo y el espacio de complejidad del problema de comparación, ya que este algoritmo se fundamenta en la metodología "divide y vencerás". Por otro lado, aprovechando las herramientas de auditoría que disponen la mayoría de los productos gestores de base de datos (DBMS), es posible obtener información valiosa de las transacciones realizadas en la base de datos. La propuesta se encuentra actualmente en desarrollo.

- Una arquitectura desde el enfoque de un sistema de detección de intrusión de base de datos, DIDS (Database Intrusion Detection System) es propuesta por (Zhang*, et al.*, 2009). Mediante una estrategia basada en reconocimiento de patrones, un mecanismo de monitorización y un proceso de auditoría del contenido de la comunicación se plantea detectar ataques de inyección SQL. Aplicando un conjunto de estrategias de detección sobre los paquetes capturados en la comunicación entre el cliente y la base de datos, y manejando una base de conocimiento con patrones de intrusiones conocidas, así como reglas de auditoría, se construye un mecanismo para el bloqueo de los ataques de inyección SQL. La eficacia en el enfoque propuesto depende de la calidad del conjunto de firmas de ataque disponible.

- Basado en la detección de usos indebidos del sistema (*misuse detection*), un enfoque de detección de intrusión es propuesto por (Asmawi, 2008). SIIMDS (*Insider Misuse Detection System*) incluye un módulo de detección de firmas, un modulo de detección de anomalía y un modulo de respuesta. Las consultas capturadas son enviadas a un motor de comparación de firmas y en el caso que no se detecte patrones de ataque, son enviadas a un motor de detección de anomalía para buscar desviaciones en el comportamiento normal de acceso a la base de datos. Si se detecta una intrusión ésta se canaliza a través del módulo de respuesta para tomar las acciones requeridas. El enfoque contempla abordar intrusiones tanto a nivel interno como a nivel externo. El enfoque se ha presentado a nivel descriptivo.

- Un enfoque basado en especificación bajo el nombre de SQL-IDS (*SQL-Injection Detection System*) fue propuesto por (Kemalis y Tzouramanis, 2008). El nuevo enfoque utiliza especificaciones de seguridad para capturar

la estructura sintáctica proyectada de las consultas SQL generadas por las aplicaciones. Capturando y guardando la estructura proyectada de los comandos SQL bajo condiciones normales de la aplicación, permite en una fase de verificación identificar las consultas maliciosas con comandos SQL inyectados. La principal limitación de este enfoque es la sobrecarga computacional para comparar en tiempo de ejecución la nueva consulta con la estructura predefinida.

- Utilizando modelos de distribuciones de caracteres, extraída desde las solicitudes HTTP, se planteó un enfoque de detección basado en anomalía. El enfoque propuesto por (Kiani*, et al.*, 2008) aborda dos tipos de ataques de inyección SQL: Tautología y el ataque basado en el operador *UNION*. Mediante un análisis sintáctico sobre la cadena SQL de la consulta, se extraen los datos de la solicitud HTTP mediante un analizador. Estos datos extraídos son usados en la fase de entrenamiento para determinar el umbral a utilizar en la fase de evaluación. El enfoque depende que el archivo de entrenamiento esté libre de ataques para una correcta detección. Adicional, la selección del umbral depende de la habilidad del administrador para escoger el valor que mejor se adapta a sus requerimientos.

- Un esquema de detección basado en anomalía aplicando de técnicas de minería de datos fue propuesto por (Bertino*, et al.*, 2007). La clave del enfoque consiste en crear perfiles de las aplicaciones con acceso a la base de datos utilizando el archivo *Log* de la base de datos. Mediante la aplicación de algoritmos de reglas de asociación se busca descubrir las reglas que representen claramente el comportamiento normal de las aplicaciones. Para cada nueva consulta bajo detección se observa si la relación entre los atributos puede ser inferida por alguna de las reglas extraídas previamente. En el caso que no se encuentre ninguna regla que se asocie a la nueva consulta, se lanza una alarma para bloquear la consulta maliciosa. Aunque el enfoque resulta interesante en su mecánica de detección, presenta el problema de encontrar un umbral adecuado para mantener una tasa baja de falsos positivos y falsos negativos.

- El problema de la detección de ataques de inyección SQL fue transformado a un problema de predicción de serie temporal utilizando redes neuronales recurrentes. El enfoque, propuesto por (Skaruz y Seredynski, 2007), se basa en la observación y recolección de secuencias de elementos (*tokens*) libre de ataques. Estas secuencias de elementos son pasados a dos tipos de redes neuronales recurrentes para ser aprendidos en una fase de entrenamiento, y luego en una fase de evaluación éstas sean capaces de predecir el siguiente elemento esperado en condiciones normales. Durante la fase de evaluación, si la nueva consulta SQL ha sido objeto de ataque, los elementos extraídos de la cadena SQL generaran un elemento diferente al esperado, detectándose el ataque.

- Un enfoque que direcciona las limitaciones de los enfoques de detección de intrusión basado en anomalía fue propuesto por (Robertson*, et al.*, 2006). El enfoque presentado utiliza técnicas de generalización para transformar las solicitudes sospechosas dentro de firmas de anomalía. Estas firmas luego son usadas para agrupar solicitudes maliciosas que presentan características similares. Otra de las técnicas utilizadas es la caracterización; mediante esta técnica permite inferir la clase de ataque asociado a la petición maliciosa. El enfoque en un principio genera una baja sobrecarga computacional, sin embargo es susceptible a generar falsos positivos.

- Un enfoque de detección de intrusión basado en arboles de decisiones para detectar ataques dirigidos a las aplicaciones Web, incluyendo ataques de inyección SQL fue planteado por (García*, et al.*, 2006). El enfoque utiliza el algoritmo ID3 para llevar a la clasificación de las peticiones. En una primera fase, las consultas SQL capturadas son sometidas a un pre-procesamiento para recuperar los atributos importantes de la cadena SQL. Con un conjunto de entrenamiento creado previamente, el algoritmo ID3 es entrenado para luego, en modo de detección, filtrar los ataques dentro de las cadenas SQL de las consultas enviadas. El enfoque presenta un porcentaje significativo de clasificaciones incorrectas.

- Otro de los enfoques dentro de la categoría de sistemas de detección de intrusión fue propuesto por (Valeur*, et al.*, 2005). Basado en el enfoque de detección por anomalía, el enfoque es entrenado usando un conjunto de consultas recuperadas desde las aplicaciones. Con las consultas recuperadas se construye una serie de modelos y en tiempo de ejecución se monitorizan las aplicaciones para identificar consultas que no se asocian con dichos modelos. La efectividad del IDS depende de la calidad del conjunto de entrenamiento.

- Un enfoque de detección basado en firmas para detectar acceso ilegal a la base de datos fue propuesto por (Low*, et al.*, 2002). DIDAFIT (*Detection of Intrusions in DAtabases through FIngerprinting Transactions*) usa una técnica para resumir las sentencias SQL dentro de un conjunto de expresiones regulares (*fingerprints*) de manera compacta. El sistema detecta una intrusión cuando ejecuta un tipo de comparación de la nueva consulta SQL contra el conjunto de expresiones regulares extraídas de las transacciones legítimas. La efectividad depende de un conjunto de expresiones regulares actualizado para detectar nuevos patrones de ataque.

2.3.2 Ataques XDoS

Hace unos pocos años la Web estaba diseñada exclusivamente para que los usuarios accedieran de manera interactiva a documentos HTML y sencillas aplicaciones; sin embargo el panorama ha ido cambiando y en la actualidad la tendencia va encaminada a dar soporte a la comunicación entre las aplicaciones para lograr una verdadera interoperabilidad entre los sistemas. Para alcanzar esta interoperabilidad entre las aplicaciones, una de las apuestas prometedoras se basa en entornos orientados a servicios (SOA, *Service-oriented architecture*). SOA es un paradigma de computación que enfatiza el descubrimiento de servicios de una manera dinámica, mediante composición y alta interoperabilidad (Singhal*, et al.*, 2007). Los servicios Web son la tecnología más prominente para hacer posible la implementación de una arquitectura orientada a servicios (Pulier y Taylor, 2005); o dicho de una forma más clara, los servicios Web se han convertido en la tecnología más efectiva para crear SOA. Existen múltiples definiciones para los Servicios Web, lo que muestra su complejidad a la hora de dar una definición consistente que abarque todo lo que son e implican. No obstante, en este trabajo se ha adoptado una definición bastante completa y aceptada de lo es un servicio Web; la definición ha sido planteada en (Cerami, 2002): Un servicio Web es cualquier servicio que está disponible sobre Internet, utiliza XML para la codificación del mensaje y no está asociado a ninguna plataforma de sistema operativo o lenguaje de programación. Una de las más importantes ventajas de usar servicios Web en el desarrollo de computación distribuida es la universalidad de las interfaces. Desde que los servicios Web pueden enviar y recibir mensajes sobre protocolos de Internet, es posible inter-operar con cualquier servicio usando simplemente cualquier computador.

Con la implementación de los servicios web, la clave está en adaptarse a una web centrada en aplicaciones, donde las conversaciones se llevan a cabo directamente entre aplicaciones similares a como se ha venido realizando entre un navegador web y un servidor. El número de áreas donde los servicios Web pueden aportar toda su potencialidad es verdaderamente amplía. Algunos de los ejemplos de primera mano que se pueden mencionar incluyen: verificación de tarjetas de crédito, seguimiento de envíos, seguimiento de la cartera, tasa de cambio, robots virtuales para la búsqueda y compra de artículos, traducción de idiomas, entre otras muchas.

Una arquitectura de servicios Web está compuesta por una pila de protocolos. Esta pila ha ido evolucionando y actualmente está conformada por 4 capas importantes (Cerami, 2002). Describiremos brevemente cada una de estas capas:

- Servicio de Transporte: Es la responsable de transportar los mensajes entre las aplicaciones de red y los protocolos. Actualmente está capa incluye el

Protocolo de Transferencia de Hiper-Texto (HTTP, *HyperText Transfer Protocol*), Protocolo Simple de Transferencia de Correo (SMTP, *Simple Mail Transfer Protocol*), Protocolo para la Transferencia de Ficheros (FTP, *File Transfer Protocol)*, y nuevos protocolos como el Protocolo de Intercambio de Bloques Extensible (*Blocks Extensible Exchange Protocol*, BEEP).

- Mensajería XML: Es la capa responsable de codificar los mensajes en un formato XML común, de forma que pueda ser entendido en ambos extremos. Actualmente, esta capa incluye XML-RPC y SOAP y REST.

- Descripción de Servicio: Es la capa responsable de describir la interface pública de un servicio Web específico. Actualmente la descripción de los servicios se maneja mediante un lenguaje de descripción de servicios Web (WSDL, *Web Services Descripción Language*).

- Descubrimiento de Servicios: Es la responsable de centralizar los servicios en un registro común y provee las funcionalidades para publicar y localizar servicios. El proceso de descubrimiento de servicio actualmente es llevado mediante la API *Universal Description, Discovery, and Integration* (UDDI).

En la medida que los servicios Web vayan evolucionando, se requerirán nuevos cambios a nivel de la pila de protocolos que conocemos actualmente. La Figura 2.6 presenta un resumen de la pila de protocolos de los servicios Web como se maneja actualmente.

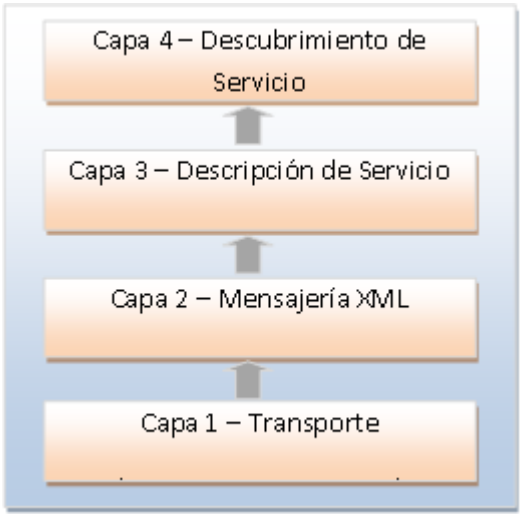

Figura 2.6. Pila de protocolos de los servicios Web

Dentro de los componentes de la tecnología de servicios Web, la transmisión de los mensajes seguramente es uno de los aspectos más importantes. El protocolo estándar SOAP (*The Simple Object Access Protocol*), es la pieza clave dentro de la arquitectura de servicios Web (Newcomer y Lomow, 2004). Está basado en XML para el intercambio de información entre computadoras; es el protocolo de comunicación que define como los mensajes deben ser construidos, transmitidos de un nodo a otro y procesado por cada nodo en su ruta hasta alcanzar el nodo destino. SOAP asume que cada mensaje tiene un remitente y un receptor, y un número arbitrario de intermediarios (nodos) que procesan el mensaje y redirigen éste al nodo destino o receptor. Aunque SOAP puede funcionar en una variedad de protocolos de transportes, HTTP es el protocolo de transporte preferido. SOAP vía HTTP realiza un intercambio de mensajes basado en el modelo Solicitud/Respuesta semejante a una Llamada a Procedimiento Remoto (RPC).

La unidad básica de comunicación entre los nodos SOAP son los mensajes SOAP. Estos mensajes están concebidos como sobres donde la aplicación encierra los datos que se van a enviar. Los mensajes SOAP contienen 3 partes importantes. La Figura 2.7 presenta la estructura típica y contenido de un mensaje SOAP.

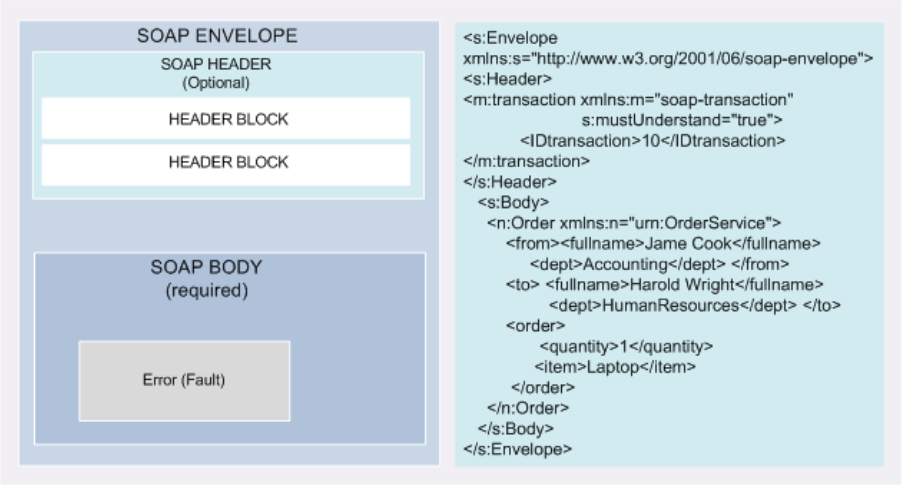

Figura 2.7. Estructura y contenido de un mensaje SOAP

- Sobre (*Envelope*): Es requerido ya que es el elemento raíz y establece el inicio y final del mensaje.
- Encabezado (*Header*): Es opcional y puede contener uno o más bloques. Permite especificar requisitos a nivel de aplicación adicionales. Usos típicos

del encabezado son: información de coordinación, identificadores (p. ej. transacciones), información de seguridad (p. ej. certificados)

- El Cuerpo (*Body*): Es requerido y puede contener uno o más bloques. Es el núcleo de la información que envía el remitente al receptor.

Con esta breve introducción de los servicios Web, se pasa al tema que nos interesa en este trabajo de tesis doctoral: los problemas de seguridad de esta tecnología. La seguridad es un elemento clave a considerar durante la fase de diseño, desarrollo e implementación principalmente en las aplicaciones consideras de alto riesgo. Las empresas han empezado a adoptar los servicios Web para dar soporte a sus aplicaciones críticas, generando una serie de inquietudes respecto a la seguridad de la información manejada y almacenada dentro de este nuevo tipo de aplicaciones.

El nuevo modelo de aplicaciones viene acompañado de desafíos de seguridad, si consideramos que las propias aplicaciones y los datos requieren atravesar enclaves corporativos de dudosa reputación. Los datos enviados en el nuevo formato de mensajería XML viajan a través de redes inseguras pasando por diferentes nodos durante su ruta hasta alcanzar el nodo destino. En su ruta, diferentes clases de usuarios y sistemas necesitan acceder a los datos para inspeccionarlos, aprobarlos y tratarlos. Si alguna parte de la cadena es comprometida, el modelo de confianza requerido se puede romper y la aplicación de negocio desarrollada como un servicio Web puede colapsar al recibir y procesar un documento XML comprometido (Rosenberg y Remy, 2004).

Las especificaciones SOAP (Gudgin, *et al.*, 2007) apuestan por una simplicidad y extensibilidad necesitando omitir características en la capa de mensajería, entre ellas la seguridad, para alcanzar estos objetivos. Estas características omitidas son frecuentemente encontradas a nivel de los sistemas distribuidos y en SOAP son dejadas para ser definidas por otras especificaciones.

Para ayudar a contrarrestar estos desafíos de seguridad, se ha estado llevando a cabo un gran esfuerzo para desarrollar estándares de seguridad dentro de los servicios Web, utilizando los medios tradicionales por parte de las agencias de estandarización en la materia. La mayoría de las tecnologías de seguridad usadas para proteger los servicios se han venido desarrollando durante años. Un ejemplo real es la criptografía, la cual se ha convertido en la columna vertebral de la seguridad en los servicios Web. Diferentes algoritmos de encriptación son aplicados para proteger la confidencialidad de los mensajes XML, la aplicación de criptografía asimétrica está siendo usada para asegurar la integridad y la firma digital de los mensajes XML, y para habilitar el intercambio de credenciales o certificados digitales entre usuarios y diferentes sistemas, se está haciendo uso de mecanismos de confianza como *Kerberos* o Infraestructuras de Clave Pública (PKI, *Public Key Infraestructure*). A esto hay que añadir, los mecanismos de seguridad ya implementados a nivel de las capas de transportes

tales como SSL/TSL (*Secure Sockets Layer/Transport Layer Security*), usados para garantizar la autenticación, confidencialidad e integridad de los datos.

Otros estándares de seguridad disponibles para su implementación dentro de los servicios Web, vienen de la mano del estándar XML. *XML Encryption* y X*ML Signature*, ambos son una Recomendación del Consorsio Web (W3C, 2009) y son dispuestos para garantizar la confidencialidad e integridad de los mensajes XML. *XML Encryption* es un lenguaje cuya función principal es asegurar la confidencialidad de partes de documentos XML a través de la encriptación parcial o total del documento transportado. *XML Encryption* se puede aplicar a cualquier recurso Web, incluyendo contenido que no es XML. *XML Signature* asegura la integridad de partes de documentos XML transportados. Además proporciona la autenticación de mensajes y/o servicios de autenticación de firma para datos de cualquier tipo, tanto si se encuentra en el XML que incluye la firma o en cualquier otra parte.

Finalmente, producto del compromiso de importantes organizaciones nace el estándar WS-Security. Actualmente bajo el desarrollo de un comité en OASIS (*Organization for the Advancement of Structured Information Standards*)(OASIS, 2009), WS-Security es un componente (*add-on*) para SOAP que describe la forma en que la cabecera de un mensaje SOAP puede ser usado para incluir información sobre la seguridad y proporcionar confidencialidad, integridad, no repudio entre otros. *WS-Security* provee mecanismos extensibles y flexibles, pero esta extensibilidad puede afectar la interoperabilidad de las distintas plataformas tecnológicas.

Según la WS-I (*Web Services Interoperability Organization*) (WS-I, 2009), los desafíos de seguridad en la transmisión mensajes SOAP incluyen:

- Identificación y Autenticación de las partes.
- Identificación y Autenticación de los datos de origen.
- Integridad de los datos: Integridad en el transporte y del mensaje SOAP.
- Confidencialidad de los datos: Confidencialidad en el transporte y del mensaje SOAP.
- Unicidad del mensaje SOAP.

Los estándares de seguridad que permiten abordar estos desafíos, los cuales son mostrados en la Tabla 2.3.

Tabla 2.3. Estándares de seguridad propuestos

• *WS-Security (Web Services Security).* • *WS-Trust* • *WS-Policy* • *WS-SecureConversation* • *WS-Federation* • *WS-Privacy* • *WS-Authorization*	• *SAML (Security Assertion Markup Language)* • *XACML (eXtensible Access Control Markup Language)* • *XKMS (XML Key Management Specification).* • *XML Encryption.* • *XML Signature*

La Tabla 2.4 muestra la relación de los estándares y los requerimientos que cumplen con ellos. La Figura 2.8 presenta gráficamente la pila de estándares.

Tabla 2.4. Relaciones entre requerimientos para seguridad en servicios Web y los estándares propuestos

Dimension	Requirement	Specification
Messaging	Confidentiality and Integrity	WS-Security
		SSL/TLS
	Authentication	WS-Security Tokens
Resource	Authorization	XACML
		XrML
		RBAC, ABC
	Privacy	EPAL
		XACML
	Accountability	None
Negociation	Registries	UDDI
		ebXML
	Semantic Discovery	SWSA
		OWL-S
	Business Contracts	ebXML
Trust	Establishment	WS-Trust
		XKMS
		X.509
	Trust Proxying	SAML
		WS-Trust
	Federation	WS-Federation
		Liberty IDFF
		Shibboleth
Security Properties	Policy	WS-Policy

	Security Policy	WS-SecurityPolicy
	Availability	WS-ReliableMessaging
		WS-Reliability

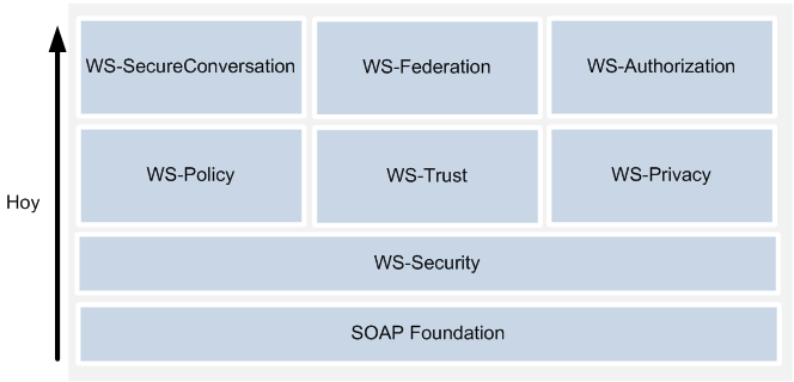

Figura 2.8. Pila de estándares de seguridad

La capa de soporte de SOAP se compone de tecnologías, como SOAP, WSDL, *XML Signature*, *XML Encryption* y SSL / TLS que son obtenidos por la especificación del WS-Security. Para conocer más detalles de cada uno de los estándares de seguridad existentes puede referirse a (W3C, 2009), (OASIS, 2009), (Rosenberg y Remy, 2004).

A pesar del esfuerzo que se ha venido realizando, la seguridad en los servicios Web sigue siendo un tema bastante complejo, principalmente provocado por la limitación que las propuestas de seguridad han aportado. Las propuestas de seguridad solo han abordado una parte del problema: garantizar la confidencial e integridad de los datos y la autenticación y autorización de los usuarios proponiendo y desarrollando una serie de especificaciones y estándares a nivel de los mensajes XML. Sin embargo, no se ha prestado atención a la disponibilidad de los servicios Web. La disponibilidad es la característica o condición que habilita a una aplicación de servicios Web continuar operando mientras sea posible bajo condiciones adversas, y pueda recuperarse y reanudar operaciones de manera normal finalizada la amenaza. El conjunto de amenazas orientadas a comprometer la capacidad de funcionamiento de los servicios va en aumento, aprovechando la inmadurez de los pocos mecanismos existentes. Esto ha originado recientemente que garantizar la disponibilidad de los servicios se convierta en uno de los desafíos más inmediato por resolver dentro de los servicios Web (Singhal, 2007). La Tabla 2.5 extraída de (Singhal*, et al.*, 2007) resume la dirección tomada por los estándares de seguridad actuales y las diferentes amenazas a considerar.

Tabla 2.5. Estándares de seguridad y amenazas a los servicios Web

Estándares	Alteración de los Mensajes (Message Alteration)	Perdida de Confidencialidad (Loss of Confidenciality)	Falsificación de Mensajes (Falsified Message)	Ataque "El hombre en el medio" (Man-in-the-Middle)	Suplantación (Principal Spoofing)	Falsificación de Petición (Forged Claims)	Replicación de partes de Mensajes (Replay of Message Parts)	Replicación de de Mensajes (Replay of Message)	Denegación de Servicio (Denial of Service)
XML-Encryption		X		X	X	X	X		
XML Signature	X		X		X	X	X	X	
WS-Security Tokens			X		X	X			
WS-Addressing								X	
SSL/TLS	X	X	X	X	X	X	X		
SSL/TLS with Client Certificates	X	X	X	X	X	X	X		
HTTP Authentication			X		X	X			

En la Tabla 2.5 se puede ver claramente que no hay estándares que protejan contra ataques de denegación de servicios, siendo precisamente los ataques DoS la principal amenaza para comprometer la disponibilidad de las aplicaciones. El ataque DoS consiste en seleccionar una víctima en particular y dirigir un número elevado de peticiones dentro de un periodo corto de tiempo. El ataque DoS se materializa cuando logra agotar los recursos dentro de la victima (Ciclos de CPU, memoria RAM, ancho de banda) (Schuba*, et al.*, 1997), (Zhao*, et al.*, 2007). Como resultado de este ataque, los usuarios autorizados no pueden acceder al servidor, interrumpiendo la prestación de los servicios.

En el caso concreto de los servicios Web, una extensión de los ataques DoS se ha convertido en la nueva amenaza para este tipo de aplicaciones. Conocido como ataque de denegación de servicio basado en XML (XML *Denial of Service Attack*, *XDoS*) (Im y Song, 2005), (Moradian y Håkansson, 2006), (Vorobiev y Han, 2006), (Gruschka y Luttenberger, 2006), este ataque toma ventaja de la incapacidad de las propuestas de seguridad existentes para direccionar esta amenaza. La probabilidad de un ataque XDoS aumenta considerablemente si tenemos en cuenta que los servicios Web están cimentados sobre una serie de estándares conocidos (Pulier y Taylor, 2005), (Cerami, 2002), incluyendo HTTP como el medio más habitual para el transporte de los mensajes y el estándar XML para la codificación de los mensajes. Los servicios Web reúnen las condiciones

idóneas para un ataque de este tipo debido a la flexibilidad nativa de los estándares, y a su naturaleza abierta.

Las técnicas de ataque XDoS a nivel de los servicios Web se aprovechan, en la mayoría de los casos, del procesamiento costoso que puede requerir ciertos tipos de peticiones de usuario. Una revisión de las diferentes técnicas de ataques XDoS contra los servicios web fue llevado a cabo por (Moradian y Håkansson, 2006). La revisión incluye mecanismos de ataques que afectan la disponibilidad de los servicios Web. En la Tabla 2.6 se presenta los tipos de ataque XDoS evaluados dentro de nuestra propuesta, tomando en cuenta el trabajo previo de Moradian y Håkansson (Moradian y Håkansson, 2006).

Tabla 2.6. Técnicas de ataque XDoS dentro de los servicios Web

	Tipo de ataque	**Descripción**	**Componente Objetivo**	**Nivel de Daño**
Evaluando estructura y contenido de los mensajes SOAP	*Recursive Payloads*	Un mensaje escrito en XML puede anidar tantos elementos haciendo la estructura compleja como para sobrecargar el analizador sintáctico requiriendo una alta demanda de memoria y recursos de procesamiento.	Analizador Sintáctico	Bajo
	Oversize Payloads	Reduce o elimina la disponibilidad de los servicios cuando un mensaje con una carga útil demasiado grande, es analizado sintácticamente dentro del servidor agotando los recursos.	Analizador Sintáctico	Bajo
	Schema Poisoning	Un atacante puede comprometer el archivo del esquema XML y reemplazarlo con otro modificado de forma maliciosa.	Analizador Sintáctico	Bajo
	Buffer overflow	Un atacante envía un valor de entrada más grande de lo esperando, causando que la aplicación no pueda manejarlo resultando en un desbordamiento de	Aplicación de Servicios Web	Bajo

		memoria.		
	XML Injection	Cualquier elemento adicionado de forma maliciosa a la estructura XML del mensaje puede llegar hasta la propia aplicación del servicio Web y causar un bloqueo.	Aplicación de Servicios Web	Bajo
	SQL Injection	Un atacante modifica la consulta SQL dentro del mensaje SOAP, adicionando comandos SQL de forma arbitraria para ejecutarlos en la base de datos.	Base de Datos	Alto
	XPath Injection	Es una forma de ataque similar a las inyecciones SQL, pero está se realiza sobre el documento XML usando XPath para extraer datos de la base de datos XML.	Base de Datos XML	Alto
Evaluando el tráfico	*Replay Attacks*	Para sobrecargar el servicio Web, el atacante intercepta los mensajes dirigidos al servicio web y luego los envía repetitivamente en periodos cortos de tiempo.	Analizador Sintáctico / Aplicación	Bajo
	XML Denial of Service attack	Un atacante inunda el servicio Web con miles de peticiones de servicio legales o maliciosas para prevenir el acceso a los usuarios legales.		Bajo

En la comunidad científica, los esfuerzos encaminados a resolver el problema de los ataque XDoS, han generado un número de propuestas con una diversidad de estrategias. A continuación se revisan los trabajos existentes en la literatura, relacionados al tema de los ataques XDoS dentro de los entornos de servicios Web.

- Una propuesta reciente basada en una arquitectura bajo el nombre de SOTA (*Service Oriented Traceback Architecture*) fue presentada por (Chonka*, et al.*, 2009). SOTA coopera con un sistema de defensa basado en un filtro, llamado *XDectector*. *XDetector* consiste en un tipo de red neuronal artificial, entrenada para detectar y filtrar mensajes maliciosos. SOTA consiste en un sistema de rastreo o seguimiento para determinar la fuente de los mensajes

maliciosos. Una vez que los ataques han sido detectados y se ha descubierto el origen, *XDetector* puede filtrar los mensajes maliciosos en camino.

- Un sistema de defensa para contrarrestar los ataques de denegación de servicios distribuidos (DDoS) y los ataques XDoS fue planteado por (Ye, 2008). El sistema ejecuta un proceso de autenticación y validación de los mensajes antes que las solicitudes sean procesadas por el proveedor. El sistema trabaja en dos modos: Modo normal y bajo ataque. En el modo bajo ataque, aquellas solicitudes que no puedan ser autenticadas y validadas son eliminadas. El enfoque permite ser reconfigurado de una forma fácil.

- Un mecanismo basado en dos estrategias fue presentado por (Srivatsa*, et al.*, 2008). La primera estrategia consiste en aplicar un control de admisión para limitar el número de clientes solicitando el mismo servicio. Este control se logra ocultando el puerto, donde el servicio recibe las peticiones, a los usuarios no autorizados. La segunda estrategia está enfocada en el control de los clientes admitidos para asignar recursos en función del estatus de los clientes. Este control es alcanzado configurando niveles de prioridad en los clientes en respuesta a sus solicitudes. La técnica presenta una baja sobrecarga en el rendimiento.

- Una plataforma adaptativa para la prevención y detección de en los entornos de servicios Web fue propuesta por (Yee*, et al.*, 2007). El enfoque plantea detectar ataques conocidos así como nuevos ataques, basándose en un enfoque hibrido que incorpora agentes, técnicas de minería de datos y lógica difusa. Los agentes actúan como sensores para detectar desviaciones de los perfiles normales aprendidos mediante la utilización de técnicas de agrupamiento, técnicas basadas en reglas secuenciales y reglas de asociación. Las desviaciones luego son analizadas usando lógica difusa para determinar ataques reales y reducir el número de falsas alarmas. Los mensajes que presentan desviaciones del perfil normal son rechazados.

- Utilizando una validación gramatical completa de los mensajes, (Gruschka y Luttenberger, 2006) proponen un sistema tipo puerta de enlace llamado *Checkway*. La estrategia de validación se lleva a cabo considerando que los mensajes están escritos en XML y que generalmente existe un archivo de esquema XML, que específica la estructura y restricciones que deben cumplir los mensajes. *Checkway* genera un esquema XML a partir de un archivo de descripción del servicio Web. Todos los mensajes enviados al servidor son validados contra el esquema manejado en *Checkway*, bloqueando los mensajes maliciosos. El enfoque presenta un modelo centralizado para detectar tipos concretos de ataque dentro de las peticiones de servicio Web.

- Basado en el diseño de un *firewall XML*, (Loh*, et al.*, 2006) proponen una solución para proteger los servicios web. La estrategia de validación está basada en un conjunto de políticas previamente configuradas. Los mensajes

SOAP enviados al servidor son capturados y procesados dentro de uno de los componentes que integran al *firewall XML*. El contenido de los mensajes es comprobado y su autenticidad validada. Los mensajes que no estén conformes a las políticas establecidas, son borrados inmediatamente. Entre las políticas configuradas incluye reglas relacionadas al tamaño de los mensajes, análisis sintáctico y validación con el esquema XML almacenado.

- Otro enfoque basado en un modelo de un firewall de servicio web bajo el nombre de *Nedgty*, fue propuesto por (Bebawy, *et al.*, 2005). El enfoque aplica reglas específicas de negocio de una manera centralizada. El enfoque trabaja a nivel de la capa de aplicación capturando los paquetes dirigidos al servidor web. El tráfico web capturado es posteriormente filtrado para recuperar los paquetes específicos de las peticiones de servicio y finalmente ejecutar una validación para detectar algún tipo de contenido malicioso. Adicional, puede filtrar solicitudes no autorizadas provenientes de direcciones IP no permitidas. Está basado en el diseño de un proxy.

- Un enfoque adaptativo presentado por (Im y Song, 2005), extiende la propuesta de (Schuba, *et al.*, 1997). Esta propuesta estaba basada en un mecanismo configurado en un firewall para monitorizar y detectar ataques de denegación de servicio del tipo inundación de paquetes SYN (*SYN flooding attacks*). Esta herramienta examina los paquetes TCP y categoriza las direcciones IP dentro de un conjunto de estados. La extensión propuesta por los autores adapta el enfoque inicial para proteger servicios Web e introduce algunas mejores consistentes en adicionar prioridad a los estados y examinar tanto los paquetes que entran y salen del servidor. Solo consideras los ataques DoS basados en inundación de paquetes SYN.

2.4 Conclusiones

El propósito de la seguridad informática es proteger los recursos valiosos de las organizaciones tales como hardware, software e información. Analizando los asuntos de seguridad y las soluciones actuales en entorno distribuidos, la tarea no resulta sencilla. Es necesario identificar qué soluciones actuales se pueden adaptar y ser aplicadas y cuáles deben ser incorporadas desde el punto de vista de los requerimientos de las nuevas aplicaciones distribuidas.

Las vulnerabilidades y amenazas en la capa de aplicación se han convertido en uno de los problemas de seguridad que están demandando mucha atención, puesto que los ataques a este nivel ponen en riesgo la confidencialidad e integridad de la información y la capacidad de prestar los servicios con la calidad necesaria a los usuarios consumidores.

La propuesta presentada en este trabajo de tesis doctoral, se centra en abordar dos de las amenazas más frecuentes y peligrosas para las aplicaciones distribuidas: Los ataques de inyección SQL y los ataques XDoS. Tomando en consideración la complejidad de estos ataques, se hace necesario evaluar nuevas soluciones de seguridad. En este sentido, la arquitectura propuesta se presenta como una solución innovadora, con un enfoque diferente al resto de soluciones existentes, para hacer frente a estos tipos de ataque.

En el capítulo 3 se describe el estado del arte de las técnicas de Inteligencia Artificial incorporadas en AIDeMaS. Finalmente, en el capítulo 4 se presenta la arquitectura AIDeMaS, describiendo en detalle cada uno de sus componentes y las innovaciones que aporta sobre el estado del arte.

Capítulo 3

Tecnologías Base

Introducción

En este capítulo se hace una revisión de la tecnología de agentes y sistemas multi-agente, el paradigma de los sistemas de razonamiento basados en casos (CBR) y técnicas y algoritmos del aprendizaje automático. Una integración adecuada de estas herramientas son la base sobre la que se fundamenta este trabajo de tesis doctoral, explotando las ventajas que cada una ofrece individualmente y fusionándolas para proporcionar una enfoque robusto y novedoso desde el punto de vista de la tecnología de los IDS.

Esta propuesta se fundamenta en la utilización de agentes deliberativos BDI, que cooperan para la resolución de distintas tareas en cada una de las etapas diseñadas para la detección de intrusión. Además, se plantea la utilización del paradigma CBR con el objeto de construir agentes dotados de una mayor capacidad de aprendizaje, adaptabilidad y autonomía. Para construir este tipo de agente se hace necesario estudiar de forma detallada los conceptos y formalismos utilizados tanto en los agentes deliberativos BDI como en los sistemas CBR.

Finalmente, como componente de clasificación, se plantea el uso de técnicas de aprendizaje automático que son incorporadas en la estructura interna de los agentes, construidos para llevar a cabo las tareas de detección de intrusiones. Las técnicas de aprendizaje automático han evolucionado hasta convertirse en la actualidad en las estrategias más aplicadas para resolver muchos de los problemas del mundo real, y que no han sido resueltas por las técnicas tradicionales. En el caso de los IDS, el aprendizaje automático proporciona numerosas técnicas y algoritmos que prometen superar las limitaciones de los enfoques IDS tradicionales. Estas técnicas proporcionan a los IDS nuevas capacidades para manejar el enorme volumen de tráfico en las redes e identificar actividades maliciosas que pueden poner en riesgo la confidencialidad e

integridad de la información, así como la disponibilidad de los servicios y recursos a los usuarios autorizados.

3.1 Tecnología de Agentes y Sistemas Multi-Agente

La teoría de agentes surge como una evolución de la inteligencia artificial distribuida (Russel y Norvig, 1995). La evolución del software, y más concretamente del software que incorpora elementos de la inteligencia artificial, tiende a la creación de entidades con comportamientos y conductas similares a las de los humanos. En definitiva, los agentes son capaces de tomar decisiones, reaccionar ante estímulos externos, cambiar su propio comportamiento y adaptarse a las necesidades del entorno.

Por otra parte, un sistema multi-agente se define como cualquier sistema compuesto de múltiples agentes autónomos con capacidades incompletas para resolver un problema global, en donde no existe un sistema de control global, los datos son descentralizados y la computación es asíncrona (Wooldridge, 2002), (Mas, 2005).

3.1.1 Concepto de Agente

Sorprendentemente la definición del término "agente" es todavía un tema de discusión, ya que no existe una definición aceptada universalmente. La razón principal se debe a que los atributos relacionados con los agentes provienen de un conjunto de disciplinadas que van desde la Psicología, la Sociología hasta las disciplinas orientadas a la computación, tales como la Inteligencia Artificial, la Ingeniería de Software y las Bases de Datos, entre otras. Estos atributos difieren de importancia en función del dominio.

A pesar de esto, diferentes autores han intentado definir lo que es un agente, es el caso de Wooldridge quién define un agente como un sistema computacional que se sitúa en algún entorno y es capaz de actuar de forma autónoma en dicho entorno para alcanzar sus objetivos de diseño (Wooldridge, 2002). Otra definición de agente desde otra perspectiva es dada por (Russel y Norvig, 1995), quien considera que un agente es cualquier cosa capaz de percibir su entorno a través de sensores y responder según su función en el mismo entorno a través de actuadores, asumiendo que cada agente puede percibir sus propias acciones y

aprender de la experiencia para definir su comportamiento. En la Figura 3.1 se muestra una representación de la anatomía de un agente.

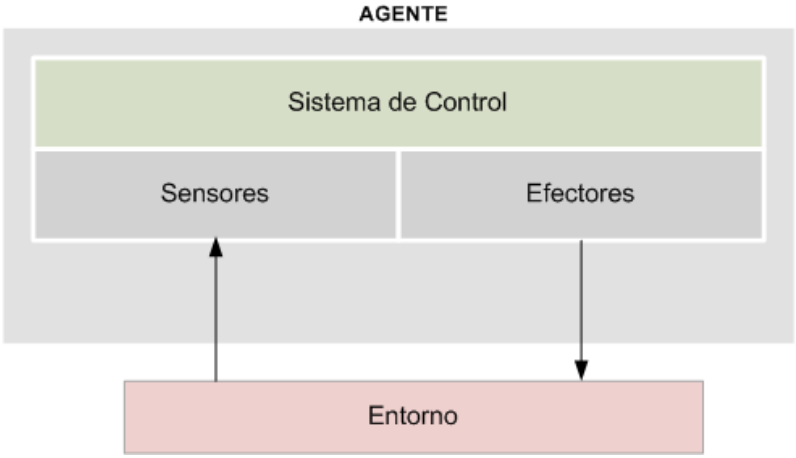

Figura 3.1. Anatomía de un agente (Jones, 2008)

Contrario a la dificultad para dar una definición de agente, existen un consenso general donde se adopta la autonomía como el concepto fundamental de agente (Wooldridge, 2002). La autonomía le permite actuar al agente sin la necesidad de intervenciones externas (humanos u otros agentes) y le provee de una clase de control sobre acciones y su estado interno. Adicional a esta característica, hay que añadir otras que los agentes deben cumplir:

- Inteligencia: Rodearse de conocimiento (creencias, deseos, intenciones y metas).
- Aprendizaje. Habilidad de adaptarse progresivamente a cambios en entornos dinámicos, mediante técnicas de aprendizaje.
- Reactividad. Percibir su entorno y actuar sobre éste con la capacidad de adaptarse a sus necesidades.
- Pro-Actividad o Racionalidad. Tomar la iniciativa para definir metas y planes que les permitan alcanzar sus objetivos.
- Movilidad: Capacidad para moverse de un sitio a otro.
- Situación. Situarse dentro de un entorno, ya sea real o virtual.
- Habilidad social. Interactuar con otros agentes, incluso con humanos.

- Organización. Organizarse dentro de sociedades que siguen unas estructuras similares a las definidas en sociedades humanas o ecológicas.

La Figura 3.2 permite esquematizar las características de un agente en relación con el entorno donde se encuentra dicho agente.

Figura 3.2. Características de los agentes de software (Botia, 2003)

3.1.1.1 Clasificación de Agentes

La clasificación de los agentes han sido abordada por distintos autores(Franklin y Graesser, 1997), (Russel y Norvig, 1995), (Brenner, *et al.*, 1998), (Maes, 1994), (Nwana, 1996) tomando en cuenta criterios como las características comunes de los agentes o de los entornos de aplicación. A continuación se presentan en la Tabla 3.1 algunas de las clasificaciones más conocidas.

Tabla 3.1. Distintas clasificaciones de los agentes.

Clasificaciones de los agentes	
Interacción con el usuario	• Agentes de interfaz. Permiten la interacción con el usuario a través de comandos. • Agentes autónomos. Aunque interactúan con el usuario, el agente decide si es necesario realizar modificaciones en el entorno debido a cambios de comportamiento del usuario.
Movilidad	• Estáticos. Se colocan dentro de un sistema o una red, siendo incapaces de realizar tareas fuera de éstos. • Móviles. Son capaces de migrar entre plataformas o entre hosts dentro una red, eligiendo de forma autónoma el momento y el destino, para una vez realizada su tarea, regresar a su origen.
Modelos biológicos	• Nivel de reino. Se dividen en robóticos y computacionales; estos últimos pueden ser agentes software o agentes de vida artificial. • Nivel de clase. Son agentes software enfocados a tareas específicas u ociosas, como los virus informáticos.
Tipo de programa utilizado para su implementación	• Reflejo simple: Actúan basándose en reglas en donde su condición concuerde con la situación actual, la cual está definida por la percepción. • Reflejo con estado interno: Mantienen información actualizada de su entorno independientemente de sus acciones y de como éstas afectan al mismo entorno. • Basados en metas: Requieren información detallada sobre las metas para elegir las acciones que le permitan alcanzarlas. • Basados en utilidad: Permiten tomar decisiones racionales cuando para satisfacer ciertas metas se presentan conflictos o si se tienen varias metas sin la certeza de lograr alguna.
Software	• De interfaz o asistentes personales. Reducen el trabajo del usuario y facilitan la interacción con el sistema. • De Internet. Se utilizan para el filtrado de información.

Sin embargo, de todas las clasificaciones en la literatura, la clasificación realizada por Nwana (Nwana, 1996) desde nuestro punto de vista es una de las más completas, siendo 7 en total los tipos de agentes, además de los agentes

heterogéneos refiriéndose aquellos sistemas integrados por distintos tipos de agentes. La Figura 3.3 presenta un gráfico de la clasificación.

- Agentes colaborativos: Son aquellos en los que se enfatizan los atributos de autonomía y cooperación.
- Agentes interfaz: Son aquellos en los que se enfatizan los atributos de autonomía y aprendizaje y que están dedicados a facilitar la interacción del usuario con el sistema.
- Agentes móviles: Se trata de agentes capaces de desplazarse a través de redes de área extensa (WANs) para efectuar sus tareas, regresando posteriormente a su origen.
- Agentes de información/de Internet: Son agentes especializados en la gestión de grandes cantidades de información de forma automática.
- Agentes reactivos: Aquellos agentes que carecen de modelos simbólicos internos y que se limitan a reaccionar ante estímulos externos.
- Agentes híbridos: Son agentes que combinan más de una filosofía o criterio.
- Agentes *smart*: Agentes inteligentes en su totalidad.

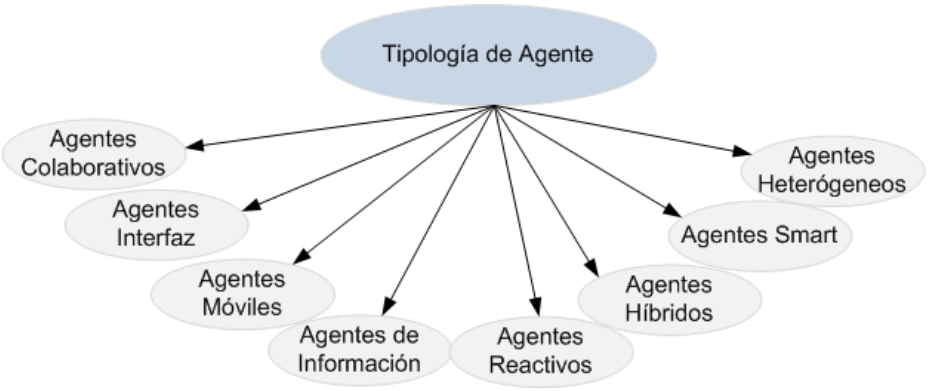

Figura 3.3. Clasificación de agentes software según Nwana (Nwana, 1996)

Es importante señalar algunos aspectos. El primero de los aspectos se refiere al tema de los atributos como criterio principal en las clasificaciones; existiendo una serie de atributos que pueden servir de referencia, como son la autonomía, la pro-actividad, el aprendizaje, y la cooperación. El segundo de los aspectos resalta el hecho que la mayoría de las clasificaciones convergen especialmente en la necesidad de diferenciar aquellos agentes que son simplemente reactivos de

aquellos que incorporan algún mecanismo de razonamiento. Y finalmente, el tercero de los aspectos hace hincapié en la existencia de agentes híbridos, que flexibilizan las clasificaciones y permiten la integración de más de un criterio de clasificación.

3.1.2 Sistemas Multi-Agente

Una vez descritos los principales requisitos que debe cumplir un agente y las características de los diferentes tipos de agentes que existen, es necesario definir lo que es un sistema multi-agente (MAS, *Multi-Agent System*). Se considera un sistema multi-agente cuando dos o más agentes son capaces de trabajar de forma conjunta con el objetivo de resolver un problema (Mas, 2005).

Dentro de la terminología de este campo es importante aclarar la diferencia entre un sistema basado en agentes y un sistema multi-agente (Jennings*, et al.*, 1998). Un sistema basado en agentes, puede contener uno o más agentes (Corchado y Molina, 2002), pero solo utiliza el concepto de agente como mecanismo de abstracción, ya que a la hora de implementarlo no existe alguna estructura de software correspondiente a éstos. Contrario a un sistema multi-agente que debe cumplir una serie de condiciones (Wooldridge, 2002):

- Al menos uno de los agentes debe de ser autónomo y debe existir al menos una relación entre dos agentes en la que se cumpla que uno de los agentes satisface el objetivos del otro. Esto quiere decir que al menos uno de los agentes dispone de información incompleta o de capacidades limitadas para resolver el problema.

- Los sistemas multi-agente se caracterizan porque no existe un sistema control global y porque cada agente se centra en su conducta individual.

- Por otro lado, los datos se encuentran organizados de forma distribuida (descentralizados), lo que favorece su computación asíncrona.

- Cada agente puede decidir con libertad, dinámicamente, que tareas debe efectuar y a quien asigna estas tareas

En un sistema multi-agente, los datos se encuentran organizados de forma distribuida y no existe un sistema de control global. De esta forma, cada agente se enfoca en su propia conducta, tomando la iniciativa guiado por sus objetivos y decidiendo dinámicamente las tareas que debe realizar o asignar a otros agentes. Por tal motivo, es necesario que los agentes trabajen de forma coordinada, principalmente a través de mecanismos de negociación, para alcanzar sus objetivos (Ossowski y García-Serrano, 1998).

Finalmente, los sistemas multi-agente han evolucionando durante los últimos años, tratando de adaptarse a los cambios que presentan las nuevas tecnologías. El papel que juega el entorno es cada vez más importante en los sistemas multi-agente (Weyns*, et al.*, 2004). Hoy en día las personas disponen de dispositivos móviles, con lo que es frecuente que un agente que se ejecuta en un dispositivo móvil cambie frecuentemente de un entorno a otro. Además, cada vez es más frecuente encontrar dispositivos inteligentes que pueden interactuar con los agentes de forma automática. Así pues, los sistemas multi-agente necesitan modelar el entorno en el que se encuentran y desarrollar mecanismos de comunicación adecuados con los elementos de dicho entorno. Esto supone la necesidad de ampliar los conceptos utilizados en las metodologías de desarrollo así como desarrollar nuevos mecanismos de comunicación y nuevas herramientas de implantación (Platon*, et al.*, 2007). Por otro lado, las tecnologías de comunicación han avanzado rápidamente, proporcionando nuevos medios de comunicación, como por ejemplo las redes inalámbricas (Fernández, 2007). Este tipo de redes introducen un alto grado de movilidad y facilitan el acceso a recursos remotos independientemente de la localización física. Este hecho hace necesario establecer nuevos mecanismos de comunicación y estandarización, que garanticen compatibilidad entre los sistemas multi-agente basados en diferentes arquitecturas. Además, la aparición de nuevos dispositivos en los que pueden ser ejecutados los agentes (Adaçal y Benner, 2006)supone la necesidad de adaptar las arquitecturas de agentes de tal forma que permitan ofrecer las mismas capacidades utilizando unas cantidades mucho menores de recursos tanto de memoria como de procesamiento.

3.1.3 Lenguaje de Comunicación entre Agentes

La comunicación es uno de los aspectos importantes dentro de una comunidad de agentes, la cual permitirá compartir y enviar mensajes entre éstos. Los métodos tradicionales permiten la comunicación entre los agentes, pero éstos no son suficientes para lograr un comportamiento social entre los agentes; para ello, es necesario que los mensajes tengan significado, es decir, un contenido semántico. Dos de los lenguajes más importantes para el envío de los mensajes son:

- KQML (*Knowledge Query and Manipulation Language*) (Fritzson*, et al.*, 1994) es uno de los lenguajes que provee la estructura para enviar mensajes entre agentes. Dentro de sus principales características se puede mencionar que es independiente de la sintaxis, del contenido y la ontología que se utilizará. Así mismo, es independiente del mecanismo de transporte (TCP/IP, SMTP, IIOP, etc.) e independiente del contenido del lenguaje (KIF, SQL, STEP, Prolog, etc.). KQML fue desarrollado por el consorcio DARPA *Knowledge Sharing*

Effort (KSE) cuyo principal objetivo fue desarrollar una base tecnológica para construir en forma progresiva sistemas cada vez con más funcionalidad, complejidad y amplitud. KQML provee un lenguaje y un protocolo para intercambiar información entre agentes inteligentes, y para lograr este intercambio se tiene que compartir una sintaxis y una semántica común para que los mensajes sean entendidos por los agentes. El lenguaje KQML se divide en tres capas. La capa de contenido alberga el contenido del mensaje, que es especificado en cualquier formato ya sea texto o binario. La capa de comunicación contiene parámetros como el identificador único asociado a la comunicación, identidad del remitente e identidad del destinatario. La capa de mensaje define tanto los diferentes tipos de interacción que se pueden establecer entre los agentes, como los protocolos que se pueden establecer entre los agentes. También, incluye otras características como el lenguaje de representación o la ontología, siendo ambos opcionales.

- ACL-FIPA: La organización internacional FIPA desarrolló un lenguaje estandarizado para la comunicación de agentes denominado *Agent Comunicación Language* FIPA (ACL-FIPA) (FIPA, 2002), cuyo principal objetivo fue darle un sentido semántico a los mensajes que intercambian los agentes. Para lograrlo se basaron principalmente en los actos comunicativos para solicitar una acción a un agente. Para la realización de una acción, el agente A envía un mensaje *request* al agente B; el receptor del mensaje podría rechazar o aceptar la acción a través de un mensaje *accept* o *refuse* respectivamente. Si el agente B aceptara la petición tiene que notificarlo e indicarle al agente A cuando finalice la realización de la acción a través del mensaje *agree*. Si en el proceso de realización de una acción se produce un fallo, se notificará al agente A mediante el mensaje *failure*.

  La estructura de un mensaje ACL-FIPA está formada por varios parámetros necesarios para la comunicación de los agentes. Un mensaje se compone de un identificador, indicando el tipo del acto comunicativo y también, incluye un conjunto de parámetros de la forma clave-valor que es la información necesaria para la realización de una acción. La estructura de un mensaje ACL-FIPA se muestra en Figura 3.4.

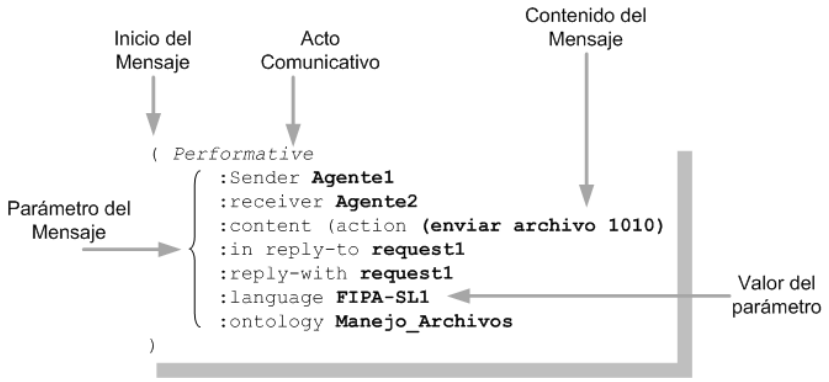

Figura 3.4. Estructura de un mensaje ACL.

3.1.4 Arquitectura de Agentes

Las arquitecturas para la construcción de agentes especifican cómo se descomponen los agentes en un conjunto de módulos que interactúan entre sí para lograr la funcionalidad requerida. Entre las principales arquitecturas tenemos las siguientes, diferenciadas en el modelo de razonamiento que utilizan:

3.1.4.1 Reactivas

Carecen de razonamiento simbólico complejo y de conocimiento o representación de su entorno, por lo que sus mecanismos de comunicación con otros agentes son muy básicos. Los agentes que utilizan este tipo de arquitectura reciben estímulos de su entorno y reaccionan ante ellos modificando sus comportamientos y el mismo entorno. Algunas de las principales arquitecturas reactivas más conocidas son:

- Arquitecturas de subsunción (*subsumption*) (Brooks, 1991) y autómatas de estado finito: Las arquitecturas de subsunción están compuestas por capas que ejecutan una determinada conducta (p. ej. explorar, evitar un obstáculo, etc.). La estructura de cada capa es la de una red de topología fija de máquinas de estados finitos. Las capas mantienen una relación de inhibición sobre las capas inferiores (inhibir entradas de los sensores y acciones en los

actuadores). El control no es central, sino dirigido por los datos en cada capa. La mayor aplicación de este tipo de arquitecturas se ha centrado en el desarrollo de controladores en robótica donde los robots se pueden considerar como agentes reales (no software) que actúan en un entorno cambiante.

- Tareas competitivas: un agente debe decidir qué tarea debe realizar de entre varias posibles, seleccionando la que proporciona un nivel de activación mayor. Se basa en una aproximación ecológica de la resolución distribuida de problemas, simulando como por ejemplo en el sistema MANTA (Drogoul, *et al.*, 1995), en el que cada agente es una hormiga y decide qué acción debe ejecutar para cumplir sus objetivos. El problema se resuelve sin comunicación entre los individuos, estableciendo un criterio de terminación del problema. Por ejemplo, los problemas clásicos de búsqueda (misioneros y caníbales, mundo de los bloques, etc.) se interpretan como agentes (cada misionero, cada bloque, etc.) que pueden realizar movimientos y se fija una condición global de terminación.

- Redes neuronales: la capacidad de aprendizaje de las redes neuronales también ha sido propuesta en algunas arquitecturas formadas por redes que son capaces de realizar una función concreta, como, por ejemplo, evitar colisiones.

3.1.4.2 Deliberativas

Utilizan modelos de representación simbólica del conocimiento basados en la planificación. Los agentes deliberativos emplean mecanismos de comunicación complejos y contienen un modelo simbólico del entorno. Toman decisiones utilizando razonamiento lógico basado en la concordancia de patrones y en la manipulación simbólica, partiendo de un estado inicial y un conjunto de planes con un objetivo a satisfacer.

A la hora de implantar una arquitectura deliberativa, hay que buscar, en primer lugar, una descripción simbólica adecuada del problema, e integrarla en el agente para que este pueda razonar y llevar a cabo las tareas encomendadas en el tiempo preestablecido (Mas, 2005).

Desde un punto de vista general, se puede decir que las arquitecturas de agentes que se estructuren siguiendo arquitecturas deliberativas de planificación, trabajarán en un espacio de decisiones desvinculado del mundo real, sobre el que posteriormente ejecutarían la acción elegida. Es decir, se trabaja con acciones posibles que se utilizan en la búsqueda de la solución a un

problema, como por ejemplo, hace cualquier sistema planificador para, tras el diseño de un plan de acción, tomar las decisiones adecuadas que le permitan alcanzar sus objetivos.

Podemos distinguir los siguientes tipos principales de arquitecturas deliberativas o simbólicas: arquitecturas intencionales y arquitecturas sociales.

- Los agentes intencionales se distinguen por ser capaces de razonar sobre sus creencias e intenciones. Se pueden considerar como sistemas de planificación que incluyen creencias e intensiones en sus planes. Entre las arquitecturas intencionales la más ampliamente extendida es la que basa su implementación en el modelo BDI (*Belif*, *Desire*, *Intention*) (Rao y Georgeff, 1995), (Bajo*, et al.*, 2006c).

- Los agentes sociales se pueden definir como agentes intencionales que mantienen además un modelo explicito de otros agentes y son capaces de razonar sobre estos modelos.

3.1.4.3 Híbridas

Para enfrentarse a entornos dinámicos y en problemas de planificación, se han propuesto algunas arquitecturas denominadas híbridas que presentan capacidades reactivas, pero que al mismo tiempo, poseen mecanismos de razonamiento deliberativo y predictivo sobre planes y objetivos.

Son arquitecturas intermedias entre las dos anteriores. Los agentes de este tipo incluyen comportamientos reactivos y deliberativos, generando un ciclo percepción-decisión-acción. El comportamiento reactivo se utiliza para reaccionar ante eventos que no requieran decisiones complejas sobre ciertas acciones.

Los más claros exponentes de estas arquitecturas son *PRS (Georgeff y Lansky, 1987)*, *TouringMachines* (Ferguson, 1992) e *INTERRAP* (Müller*, et al.*, 1994), (Müller, 1996).

Finalmente, para concluir el tema de los tipos de arquitecturas de agentes, es importante a la hora de plantearse la implementación de una arquitectura multi-agente reconocer las características del entorno. En nuestro caso, los agentes deliberativos son la mejor opción para abordar el problema de la detección de intrusiones en entornos dinámicos donde se requiere que los agentes tomen decisiones de cierta complejidad, y puedan adaptarse conforme las amenazas van evolucionando.

3.1.5 Modelo BDI (*Belief-Desire-Intention*)

El modelo BDI parte de la base filosófica de la teoría de las Creencias-Deseos-Intenciones presentada por Bratman (Bratman*, et al.*, 1988) como modelo de razonamiento práctico humano. En el modelo BDI la estructura interna de un agente y su capacidad de elección se basan en aptitudes mentales. Esto tiene la ventaja de utilizar un modelo natural y de alto nivel de abstracción.

En una arquitectura BDI (*Belief*, *Desire*, *Intention*), los agentes que la implementan están dotados de los estados mentales de Creencias, Deseos e Intenciones (Bratman, 1987). Posiblemente ha sido el modelo más difundido y el más estudiado dentro de los modelos de razonamiento de agentes principalmente porque el modelo BDI combina elementos interesantes: un modelo filosófico de razonamiento humano fácil de comprender, un número considerable de implementaciones (Georgeff y Lansky, 1987), y se ha desarrollado una semántica lógica abstracta y elegante, la cual ha sido aceptada por la comunidad científica (Georgeff*, et al.*, 1998), (Rao y Georgeff, 1998). Estas razones principalmente le han valido para ser el modelo de agente deliberativo más utilizado en la actualidad, ya que es el que más se asemeja al modelo de razonamiento humano. A continuación se describen las principales aptitudes mentales de los agentes:

- Las creencias representan la parte informacional del sistema. Son la representación, con valores o expresiones, del estado del entorno y de los estados internos del agente. Aunque representan información imperfecta, son esenciales para recordar eventos pasados y para mejorar la percepción con el entorno. Las creencias resultan aún más necesarias cuando los recursos del sistema son limitados y se deben calcular acciones recurrentes continuamente. Es posible modificar la información almacenada cuando se detecten cambios en el entorno (Rao y Georgeff, 1995).

- Los deseos (u objetivos) consisten en aptitudes motivacionales del sistema. Pueden ser simplemente el valor de una variable, un registro o una expresión, pero que representa algún estado final deseado. Normalmente, los agentes cuentan con más de un objetivo, así que el sistema debe tener información acerca de los objetivos y las prioridades de cada uno (Rao y Georgeff, 1995).

- Las intenciones constituyen la parte premeditada o deliberada del sistema. Son un conjunto de caminos de ejecución (*threads*) que pueden interrumpirse al recibir información que involucre cambios en el entorno (Kinny y Georgeff, 1991). Las intenciones permiten modificar ciertas acciones para alcanzar los objetivos. El sistema almacena las intenciones y

planes actuales para utilizarlos en situaciones futuras (Rao y Georgeff, 1995).

Los agentes deliberativos BDI se modelan utilizando una estructura abstracta, basada en la lógica de situaciones (o mundos posibles), denominada árbol temporal con múltiples futuros y un solo pasado (Rao y Georgeff, 1995). En este árbol, cada nodo es una situación, y las ramas son las opciones del agente en un momento dado. Para cada situación se definen una serie de nuevas situaciones que el agente puede alcanzar desde el punto de vista de las creencias (situaciones que se consideran posibles), de los deseos (situaciones que se desean alcanzar) y de las intenciones (situaciones que se intentan alcanzar) (Rao y Georgeff, 1995). Para este modelo es necesario que existan ciertas relaciones entre las creencias, los deseos y las intenciones del agente:

- Compatibilidad entre creencias y objetivos. Si el agente desea alcanzar un objetivo, debe creer que dicho objetivo es cierto.

- Compatibilidad entre objetivos e intenciones. Antes de que el agente se enfoque en una intención, éste debe haberla formulado como deseo.

- Las intenciones conducen a acciones. Es necesario que el agente ejecute las intenciones a través de acciones simples.

- Relación entre creencias e intenciones. El agente debe creer en sus propias intenciones.

- Relación entre creencias y objetivos. El agente debe conocer sus objetivos y deseos.

- No hay retrasos infinitos. Cuando un agente adopta una intención para alcanzar un objetivo, debe continuar hasta un determinado momento finito.

Así, las intenciones actuales del agente guían o influyen en sus decisiones sobre futuras intenciones. Dependiendo de cómo afecten las intenciones pasadas a las futuras, se identifican varios tipos de agentes (Rao y Georgeff, 1991):

- Ciego. El agente mantiene sus intenciones hasta que sabe que las ha alcanzado. Por lo tanto, si es necesario rechazará las creencias o deseos que contradigan sus compromisos.

- Firme. El agente mantiene sus intenciones mientras crea que tiene opciones de alcanzarlas.

- Imparcial. El agente mantiene sus intenciones mientras éstas se corresponden con sus deseos, es decir, mientras el deseo o deseos que dieron lugar a esa intención no cambian.

A nivel de implementación, las creencias, los deseos y las intenciones se almacenan separadamente en listas y se trabaja con secuencias de eventos en una estructura de datos lineal con comportamiento FIFO (primero en entrar primero en salir). La arquitectura BDI realiza repetidamente una serie de pasos, cada uno de ellos con una duración limitada y con posibilidad de reacción ante el entorno. Al comenzar el ciclo, se lee la cola de eventos y se devuelve una lista de opciones, en donde se seleccionan aquellas que se deben adoptar, y se añaden a la cola de intenciones. A continuación, se ejecutan todas las intenciones que impliquen la realización de una acción simple, y se comprueba si existen nuevos eventos en el entorno para incorporarlos a la cola de eventos. Por último, el agente modifica las estructuras de deseo e intención, eliminando los ya satisfechos y los que son imposibles de alcanzar (Rao y Georgeff, 1995).

La secuencia de pasos en la arquitectura abstracta y el esquema del ciclo de ejecución son presentados en la Figura 3.5.

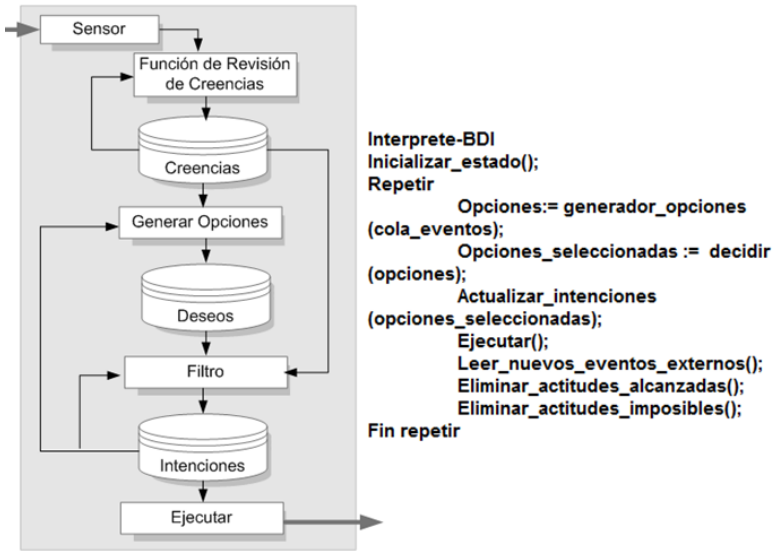

Figura 3.5. Secuencia en la arquitectura abstracta y esquema del ciclo de ejecución propuesta por Rao y Georgeff (Rao y Georgeff, 1995)

Uno de los pilares fundamentales de esta investigación es el uso de agentes y sistemas multi-agente. Esta tecnología puede proporcionar múltiples ventajas en las tareas de detección de intrusiones. Una de esas ventajas está relacionada con la capacidad de computación distribuida que ofrecen los agentes. Las tareas del proceso de detección pueden distribuirse dentro del conjunto de agentes mejorando los tiempos de respuesta y explotando los recursos disponibles. Pero principalmente, los agentes poseen habilidades que los hacen especialmente

adecuados para la detección de intrusiones, más concretamente los agentes con una arquitectura deliberativa BDI. Estos agentes poseen capacidades de razonamiento y aprendizaje que son esenciales para aprender de los ataques. La utilización de agentes deliberativos BDI es una pieza clave en el desarrollo de la arquitectura AIDeMaS, la cual se describe a detalle en el capítulo 4 de esta memoria. En la siguiente sección se describe el paradigma del razonamiento basado en casos, que puede ser integrado en los agentes BDI para aumentar sus capacidades.

3.2 Modelo de Razonamiento Basado en Casos (CBR)

El razonamiento basado en casos (CBR) es un tipo de razonamiento, utilizado en el pensamiento humano, en el que se recurre a experiencias pasadas para resolver nuevos problemas (López De Mantaras y Plaza, 1997).

El CBR es otro paradigma de resolución de problemas, pero con diferencias sustanciales con el resto de los acercamientos de la inteligencia artificial, haciéndolo atractivo para abordar diferentes tipos de problemas. Por ejemplo, los sistemas CBR en vez de confiar únicamente en el conocimiento general del dominio del problema, o llevar a cabo asociaciones a lo largo de relaciones entre descripciones del problema y conclusiones, este paradigma es capaz de utilizar conocimiento específico de experiencias pasadas, en otras palabras, situaciones de un problema concreto (casos). Ante el planteamiento de un problema no abordado con anterioridad, se intenta localizar un caso pasado similar y adaptar su solución a la situación del problema nuevo. De esta adaptación podemos obtener una nueva experiencia a la hora de resolver problemas con ciertas similitudes, lo que nos lleva a una segunda diferenciación del CBR con respecto al resto de tendencias, el aprendizaje incremental, ya que las nuevas adaptaciones se almacenan como nuevos casos, relacionados, y disponibles para comparaciones futuras.

Así pues, la aplicación de modelos CBR requiere tener en cuenta dos aspectos importantes. En primer lugar, el modelo se basa en la idea de que problemas similares tienen soluciones similares. Sin embargo, carecer de problemas similares no supone que el sistema no sea capaz de proponer buenos resultados, sino que la reutilización de memorias pasadas se convierte entonces en un proceso creativo. Sea cual sea el resultado de este proceso creativo, el individuo aprende de la nueva experiencia. En segundo lugar, en el proceso de razonamiento explicado se está emitiendo un juicio de valor que permite saber si la solución aplicada para resolver un determinado problema fue buena o no fue buena. Si este juicio es emitido por un experto en la materia del problema

resuelto, mayores serán las posibilidades de incrementar la capacidad de aprendizaje (Schank, 1983).

Debido al concepto amplio del término CBR, algunos autores como (Aamodt y Plaza, 1994) distinguen diferentes tipos o especializaciones para el razonamiento basado en casos, especialmente atendiendo a la representación, indexación o a los mecanismos de razonamiento aplicados sobre los casos:

- *Exemplar-Based Reasoning*: Según Smith y Medin (Smith y Medin, 1981) la definición de un concepto puede ser analizada desde tres puntos de vista: vista clásica, vista probabilística y vista de ejemplo. En la vista de ejemplo, cada concepto se define por extensión, como un conjunto de ejemplares. De esta forma se puede intentar aprender varias definiciones de un mismo concepto en modo ejemplar de definición. Este tipo de sistemas se utilizan en tareas de clasificación. La fase de adaptación de las soluciones al caso de estudio tratado no está presente.

- *Instance-Based Reasoning:* El razonamiento basado en instancias es una especialización del anterior, que introduce una mayor aproximación sintáctica. En este caso para definir un concepto se necesita un cierto número (elevado de instancias). Las instancias suelen tener una representación sencilla (vectores) y se intenta facilitar el aprendizaje automático (Aha*, et al.*, 1991), (Fdez-Riverola*, et al.*, 2007).

- *Memory-Based Reasoning*: En este caso, se hace énfasis en la idea que una colección de casos es una memoria de tamaño considerable. De esta forma el proceso de razonamiento es visto como un proceso de acceso y búsqueda sobre dicha memoria. En el razonamiento basado en la memoria, la forma en la que se encuentre organizada la memoria y la forma en la que se accede a la memoria son fundamentales y centran la mayor parte de los estudios y esfuerzos (Stanfill y Waltz, 1988), (Kopena y Regli, 2003). Suele utilizarse en procesamiento en paralelo.

- *Case-Based Reasoning*: Típicamente un sistema CBR propiamente dicho viene caracterizado por el concepto de caso. Cada caso dispone de abundante información sobre la situación que representa, con una organización interna un tanto compleja. Además, este tipo de sistemas son capaces de adaptarse a distintos entornos o contextos, facilitando la generalización y cierto grado de independencia del entorno (López De Mántaras, 2001).

- *Analogy-Based Reasoning:* Se trata de sistemas que utilizan la generalización por analogías para resolver problemas en un dominio determinado utilizando las experiencias obtenidas previamente en otros dominios diferentes (Veloso y Carbonell, 1993).

A continuación se dará una definición de caso y se detallarán los cuatro componentes principales de un sistema CBR.

3.2.1 Concepto de Caso

En la introducción de este paradigma, hemos hablado de razonamiento basado en casos; se ha explicado que se trata de un razonamiento basado en la experiencia, en recuerdos, y más de una vez hemos empleado el término "caso". El concepto de caso es fundamental en los sistemas CBR, tanto que la estructura de un sistema CBR se diseña sobre este concepto. Para Schank (1983) un caso es un pedazo de conocimiento contextualizado que representa una experiencia, de tal forma que un caso contiene:

- El problema que describe el estado del mundo cuando ocurrió el caso
- Una descripción de la solución encontrada y/o
- Un resultado que describe el estado del mundo después de que ocurrió el caso.

De esta forma, un caso se puede ver como una lección aprendida cuando se ha resuelto un determinado problema. Los casos pueden representarse de diversas formas, como instancias, como vectores, como objetos, etc. Un caso se representa por medio de características llamadas índices. Los índices podrán ser restricciones, objetivos, solución, fracasos, etc.

Para Kolodner (1993) un caso es una porción de conocimiento contextualizada que representa una experiencia y que sirve como base fundamental para alcanzar las metas marcadas para el problema actual. Así, para Kolodner (1993), un caso se compone de una descripción del problema, de la solución para ese problema y del resultado obtenido tras aplicar la solución. De esta forma se puede utilizar una notación matemática en la que un caso se representa por medio de una 3-tupla, *<P, S(P), R>*, como se muestra a continuación.

Case: <Problem, Solution, Result>

Problem: initial_state

Solution: sequence of <action, [intermediate_state]>

Result: final_state

No existe un consenso en cuanto a la información que debe aparecer representada en un caso, sin embargo, se debe de considerar la funcionalidad y la facilidad de adquisición de la información representada en el caso. Así pues, un caso queda definido a través de tres elementos:

- *P* o Problema: Representa una descripción del problema que se desea resolver.

- *S(P)* o Solución: Es la solución del problema *P*, y viene representada a partir del conjunto de operadores que se utilizan para construir la solución.

- *R* o Resultado: Es la eficiencia, que mide los recursos utilizados para alcanzar la solución.

El enfoque computacional del CBR trata de trasladar el modelo de pensamiento humano basado en la experiencia al mundo de la inteligencia artificial. Así pues, se plantea y estudia todos aquellos elementos que pueden ser necesarios para implementar un modelo CBR de forma artificial. Riesbeck y Schank (1989) proponen una estructura que permite modelar un sistema CBR, de tal forma que se describen los componentes principales del sistema y las relaciones que se establecen entre ellos. El sistema debe estar formado por dos componentes fundamentales: la memoria de casos y el mecanismo de razonamiento (que a su vez puede necesitar de la inclusión de una base de conocimiento).

3.2.2 Ciclo de Vida de un Sistema CBR

La resolución de problemas en un sistema CBR se lleva a cabo mediante la ejecución de una serie de pasos conocida como el ciclo de vida de un sistema CBR o ciclo CBR. En el ciclo CBR se especifican los pasos ordenados y relacionados por tiempo mediante los cuales se extrae y aprende información para resolver un problema específico. Según Aamodt y Plaza (1994), el ciclo CBR está formado por cuatro procesos secuenciales. Estos cuatro procesos son conocidos como las cuatro *Rs*: *Retrieve* (Recuperación), *Reuse* (Adaptación), *Revise* (Revisión) y *Retain* (Retención o aprendizaje) (Lopez De Mantaras*, et al.*, 2005).

Ante un nuevo problema, el sistema recurre a la experiencia almacenada para obtener la solución más adecuada. Así pues, lo primero que hace el sistema CBR es ejecutar la fase de recuperación. Se trata de una fase en la que se busca en la memoria aquellos casos con una descripción de problema más similar al problema actual. Seguidamente se ejecuta la fase de adaptación, en la que el sistema trabaja con las soluciones correspondientes a los casos más similares recuperados en la fase anterior. El resultado de la fase de adaptación es una solución al problema actual. La solución propuesta se revisa o evalúa, comprobando su validez. Finalmente, en la fase de aprendizaje, el sistema almacena la nueva experiencia y aprende de ella. En algunas aplicaciones prácticas, la etapa de reutilización y revisión son difíciles de diferenciar y en ocasiones se implementa una fase de adaptación que reemplaza y combina ambas fases (Shiu y Pal, 2004). En cada una de las fases del ciclo CBR se puede aplicar distintas técnicas y algoritmos. Por ejemplo, en la recuperación de casos

similares se pueden aplicar diferentes técnicas y algoritmos de similitud. Opcionalmente se puede utilizar una fase de revisión del conocimiento experto.

Con el ciclo CBR propuesto por Aadmodt y Plaza (1994) se puede comprobar cómo el ciclo se inicia con la llegada de un nuevo problema y la obtención del conjunto de caso correspondiente. El sistema identifica las características más relevantes del problema planteado y las plasma en la estructura correspondiente a la descripción de problema de un caso. Una vez que se ha identificado el conjunto de caso el sistema pasa a resolverlo y, para ello, ejecuta el ciclo CBR etapa por etapa. En la Figura 3.6 se presenta gráficamente el ciclo de vida de un sistema CBR.

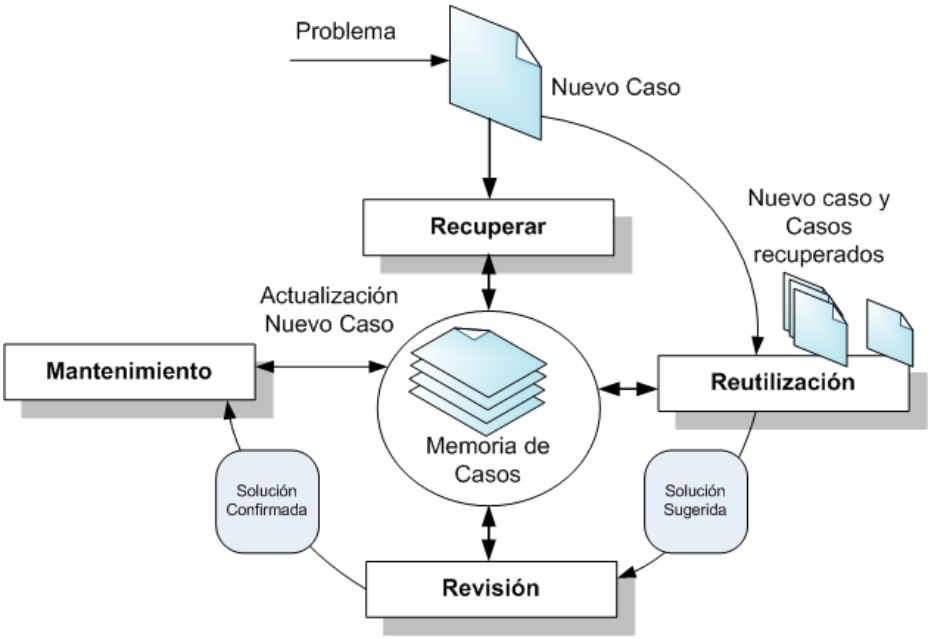

Figura 3.6. Ciclo CBR (Aamodt y Plaza, 1994)

- Fase de recuperación de los casos (*retrieval*): Es la primera etapa que realiza un sistema CBR. En esta etapa se realiza la recuperación de casos, esto es, el acceso a los casos almacenados que cuentan con una descripción de problema más similar a la del problema actual. En esta etapa se llevan a cabo dos funciones distintas: Acceso a los casos almacenados y establecer la similitud entre casos (más concretamente entre descripciones de problema). De esta forma es necesario plantearse:

- Algoritmo de acceso a los casos almacenados: Preferiblemente debe tratarse de un algoritmo que garantice un acceso rápido y eficiente. El algoritmo tendrá una alta dependencia de la organización que se decida utilizar en la memoria de casos.

- Técnicas o métricas que permitan determinar la similitud entre casos: A aquellos casos a los que se accede mediante el algoritmo de acceso se les aplica una métrica de similitud que permita determinar cuál o cuáles de ellos son los mejores casos (los más similares al problema actual) (Golding y Rosenbloom, 1988). Obviamente es posible aplicar diferentes técnicas y utilizar distintos criterios para decidir cuál es el caso más similar. La métrica de similitud tiene en cuenta las distintas características que permiten definir un problema y a cada una de ellas le asigna un determinado peso.

- Fase de Reutilización de los casos (*reuse*): Una vez finalizada la etapa de recuperación, el sistema CBR pasa a ejecutar la etapa de reutilización. Esta etapa recibe como entradas los casos más similares recuperados durante la primera etapa. La reutilización o adaptación consiste en trabajar con las soluciones correspondientes a los casos más similares recuperados en la etapa de recuperación para obtener una solución aplicada al problema actual. Trabajar con las soluciones significa modificarlas y combinarlas, o simplemente decidir cuál de ellas es la más óptima y reutilizarla.

  En la fase de adaptación existen muchas posibilidades para llevarla a cabo. Lo ideal sería encontrar un caso con una descripción de problema idéntica al actual. Pero, dado que en el mundo real es muy complicado que se presenten dos situaciones idénticas, se hace necesario crear una nueva solución basándose en las soluciones similares de las que se dispone. Además de los datos proporcionados por los casos similares recuperados se hace necesario utilizar algún tipo de conocimiento. El conocimiento puede venir dado a través de fórmulas o reglas.

- Fase de revisión de la solución: (*Revise*): Una vez que la solución considerada apropiada ha sido generada, se comprueba si la solución propuesta es apropiada para el caso actual. Para ello se utiliza un sistema experto de conocimiento, o bien una persona experta. El resultado de esta etapa es un nuevo caso, para el que se haya obtenido una solución satisfactoria o una solución incorrecta y deba ser reparado. En ocasiones, en esta etapa se puede realizar una reparación de los fallos o errores detectados.

- Fase de Aprendizaje y Mantenimiento (*Retain*): En esta última etapa se realiza la fase de aprendizaje donde el sistema adquiere nuevo conocimiento a partir de la nueva experiencia. El proceso se lleva a cabo almacenando el caso actual y la solución aplicada para resolverlo. Además se tiene en cuenta el resultado obtenido en la etapa de revisión para asignar una eficiencia al caso. De esta forma el caso puede ser indexado en la memoria de casos. Un

elemento importante es el mantenimiento de la memoria de casos que puede crecer rápidamente; para ello puede ser necesario reorganizar la memoria de casos. En caso de similitud con otros casos se puede aplicar generalización. En ocasiones durante esta etapa, o bien como una etapa adicional previa a la retención y aprendizaje se incluye una revisión de la base de conocimiento. En el caso de que se estén utilizando reglas u otro tipo de conocimiento experto, el conocimiento es revisado y actualizado en función de los resultados obtenidos para la experiencia que se acaba de vivir.

3.2.3 Memoria de Casos

La memoria de casos es la encargada de mantener la representación y organización de los casos. La base de casos debe tener en cuenta la estructura de los casos (representación de los casos), y debe tratar de facilitar en la mayor medida de lo posible cada una de las operaciones que se realizan en el ciclo CBR. Para facilitar dichas operaciones es necesario contemplar tanto la indexación y organización de los casos (asignación de índices para facilitar su recuperación) como el mantenimiento (Leake*, et al.*, 2000).

- Representación de los casos: La representación de un caso debe ser tal que facilite en la mayor manera posible las etapas del ciclo CBR. Por ejemplo, el proceso que propone Leake (Leake*, et al.*, 1996), permite definir un objeto para la descripción de problema y otro objeto para la solución correspondiente. Dos de los modelos propuestos (Leake*, et al.*, 1996) para la representación de casos son el modelo de memoria dinámica basada en una estructura jerárquica denominada EMOPs (*Episodic Memory Organization Packets*). La idea básica consiste en organizar casos específicos que comparten propiedades similares en una estructura más general (en lo que ellos llaman un episodio generalizado). El segundo es el modelo de categoría-ejemplar (Porter*, et al.*, 1990) donde los casos o ejemplares, y su fundamento se encuentra en la propiedad de extensibilidad. Se define una estructura en la que intervienen categorías, relaciones semánticas, casos y punteros a índices. Cada caso se asocia con una categoría. Un índice puede apuntar a un caso o a una categoría. Adicional a los modelos mencionados, otros modelos propuestos tienden a incorporar tendencias más actuales como la de la representación mediante objetos (Bergmann y Stahl, 1998) y las tendencias de estandarización para la representación de la información (Coyle*, et al.*, 2004).

- Indexación de la memoria de casos: Consiste en la utilización de índices que faciliten el acceso a los casos almacenados. Si no se utiliza indexación, el proceso de recuperación puede ser muy costoso. Evidentemente la

indexación va a estar muy relacionada con la representación que se utilice para los casos. Los índices son características de los casos que permiten distinguirlos de otros casos. Generalmente, atendiendo a estos índices se crean jerarquías de casos (grupos o categorías). Así pues, la indexación utilizada es una parte crucial de un sistema CBR, ya que determina las circunstancias en las que los casos van a ser recuperados (Kolodner, 1993). Según Riesbeck y Schank (1989) un buen índice es aquel que es distintivo pero no único. La estructura piramidal donde los casos se estructuran en jerarquías en base a algún índice o conjunto de índices es la forma más frecuente para organizar los casos.

- Mantenimiento de la memoria de casos: Dado que un sistema CBR está en continua ejecución, el tamaño de la memoria de casos crece constantemente. La ejecución del sistema CBR se verá afectada por el incremento en la complejidad del sistema y por la utilización de casos redundantes (Leake, 1998), (Ashley y Brüninghaus, 2003). El mantenimiento de la memoria de casos se centra en cómo reducir el tamaño de la memoria de casos manteniendo una alta eficiencia en la solución de problemas. Según (Zhi-Wei*, et al.*, 2005) los métodos más utilizados para realizar el mantenimiento de la memoria de casos hoy en día son el ajuste dinámico de la estructura de la memoria de casos (Racine y Yang, 1997), *clustering* jerárquico, ajuste de índices de la memoria de casos, definición de reglas para el mantenimiento, *data mining*, etc. (Zhi-Wei*, et al.*, 2005). Adicional, otras estrategias propuestas incluyen la utilización de métricas que permitan detectar casos redundantes e inconsistentes (Leake, 1998), IB3 (métodos de borrado preservando la competencia) (Smyth y Keane, 1995), borrado de casos (Racine y Yang, 1997), detección de agentes (Minton, 1990). A pesar de los múltiples enfoques propuestos, hoy en día el mantenimiento de la memoria de casos es un problema para el que todavía se buscan soluciones óptimas.

Finalmente para concluir el tema de la memoria de casos, un componente que ya se mencionó en la etapa de mantenimiento del ciclo CBR es la posibilidad de manejar una base de conocimiento la cual contendrá informaciones tales como teorías, principios, reglas, etc. que permitan tomar decisiones para la solución de problemas y aprender de las experiencias. Se trata de conocimientos que permiten realizar funciones tales como la generalización, la toma de decisiones, etc. con lo cual son funciones complicadas en las que el mantenimiento también juega un papel muy importante. El conocimiento del que dispone el sistema puede ser un sistema experto. El sistema de conocimiento experto debe ser capaz de aprender de las experiencias adquiridas por el sistema CBR. Probablemente se trata de la parte más complicada de un sistema CBR.

3.2.4 Ventajas y Desventajas en la Aplicación de Modelos CBR

En los últimos años, el Razonamiento Basado en casos (CBR) ha experimentado un rápido crecimiento desde su nacimiento y se ha convertido en un enfoque de amplio interés, multidisciplinar y de gran interés comercial.

Las principales ventajas del uso de CBR desde diferentes puntos de vistas son mencionadas a continuación (Shiu y Pal, 2004).

- Reducir la tarea de adquisición de conocimiento: Las recolección de casos/experiencias existente y relevante, su representación y almacenamiento son las principales tareas a diferencia de otros modelos que requieren extraer el conocimiento a partir de modelos o conjuntos de reglas.

- Evitar repetir errores hechos en el pasado: Los sistemas que realizan un registro detallado de las situaciones que se han ejecutado correctamente y las situaciones que han producido fallos, permiten reutilizar este conocimiento para predecir futuros fallos.

- Provee flexibilidad a la hora de modelar el conocimiento: El uso de experiencia pasada como dominio de conocimiento y la capacidad de proveer soluciones razonables aplicando estrategia de adaptabilidad apropiadas permite la solución de problemas complejos.

- Razonamiento en dominios que no han sido completamente entendidos, definidos o modelados: Un razonador de caso puede funcionar adecuadamente con un pequeño conjunto de casos del dominio.

- Hacer predicciones sobre el éxito probable de una solución ofrecida: Con la información almacenada (experiencia previa), el razonador puede ser capaz de predecir el éxito de la solución para el problema actual.

- Aprendizaje sobre el tiempo: Aquellos casos resueltos y probados en el mundo real de forma exitosa son adicionados a la base de casos y usados para resolver problemas futuros.

- Razonamiento con datos y conceptos incompletos e imprecisos: Los casos recuperados puedan que no sean necesariamente idénticos al caso actual, pero si caen dentro de alguna medida de similaridad, el razonador puede manejar las diferencias y continuar razonando.

- Evitar repetir todos los pasos requeridos para llegar a una solución: En los problemas que requieren un proceso costoso para crear una solución desde cero, un enfoque alternativo sería modificar una solución anterior o

reutilizar una solución previa para reducir de una manera significativa dicho proceso.

- Proveer un medio de explicación: Los sistemas CBR pueden ayudar a convencer a un usuario o justificar la solución propuesta al nuevo caso. Utilizando el proceso de un caso anterior hasta llegar a la solución, se puede utilizar para proveer las explicaciones necesarias.

- Aplicable a muchos propósitos y rango de aplicaciones diferentes: Los sistemas CBR pueden ser aplicados a multitud de propósitos tales como crear planes, hacer diagnósticos, clasificaciones, debatir puntos de vistas, etc.

- Reflejan el razonamiento humano: En muchas situaciones, las personas usan formas de razonamiento basados en casos, lo cual permite de una forma más fácil convencer de la validez del paradigma y la validez de las soluciones ofrecidas por los sistemas CBR.

La aplicación de enfoques CBR también presenta algunas desventajas que deben ser consideradas.

- Existe la posibilidad de caer en una tendencia a usar los casos previos ciegamente, confiando en la experiencia previa sin validarla con respecto a la nueva situación.

- Dependiendo de la estrategia del razonador, es posible que los casos previos pueden predisponer demasiado al razonador a la hora de resolver el nuevo problema.

- Lo más típico en un sistema CBR es que en el arranque normalmente no se disponga del conjunto de casos más apropiado para el tratamiento de un problema concreto.

- La validación de la experiencia previa juega un papel importante para evitar evaluaciones ineficientes incorrectas. La recuperación de casos inapropiados puede costar un tiempo considerable o llevar a errores muy costosos, que podrían ser evitados por métodos más incrementales.

Los sistemas CBR pueden resultar adecuados para afrontar el problema de detección de ataques de inyección. La continua evolución de los ataques requiere de una estrategia que permita al mecanismo de detección adaptarse a los cambios que se producen en los patrones de ataque. En esta dirección, los sistemas CBR proporcionan la capacidad de aprender a través de la experiencia. En el problema de los ataques de inyección los sistemas CBR aportan una capacidad de aprendizaje incremental a medida que adquieren experiencia. La nueva experiencia obtenida durante el proceso de solución de problemas pasados permitirá resolver nuevos problemas con características similares. Esto

puede permitir asegurar una capacidad de adaptación continua para hacer frente a las diferentes técnicas de ataques de las amenazas estudiadas.

3.3 Aprendizaje Automático y Minería de Datos Aplicado a la Detección de Intrusión

La detección de intrusión es una de las tecnologías claves de la seguridad informática centrada en la identificación de actividades maliciosas, siendo la estrategia de detección basada en usos indebidos (firmas de ataque) el enfoque más aplicado. Sin embargo, como ya se ha mencionado anteriormente, su limitación principal está en el mantenimiento costoso de la base de datos de firmas para mantener la efectividad en la detección. Para superar esta y otras limitaciones, la aplicación de técnicas de minería de datos y aprendizaje automático se han convertido en la nueva línea de investigación en los IDS. Tanto la minería de datos como el aprendizaje automático construyen modelos a partir de un conjunto de datos de ejemplo. La minería de datos es el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos (Witten y Frank, 2000), mientras que el aprendizaje automático se ocupa de desarrollar algoritmos capaces de aprender, y constituye, junto con la estadística, el corazón del análisis inteligente de los datos. El modelo construido puede ser predictivo para hacer predicciones en el futuro, descriptivo para ganar conocimiento de los datos, o sencillamente la aplicación de ambos (Alpaydin, 2004). Los principios seguidos en el aprendizaje automático y en la minería de datos son los mismos: la máquina aprende un modelo a partir de ejemplos y los usa para resolver el problema.

Una característica principal dentro de las técnicas de aprendizaje es el paradigma de aprendizaje de los sistemas. En el aprendizaje supervisado, se realiza la estimación basada en un conjunto de entrenamiento, que incluye la clase a la que pertenecen los ataques. Dicha estimación se extrae mediante la utilización de diversos algoritmos sobre un conjunto de registros de ataques previamente etiquetados. En cambio, en el aprendizaje no supervisado, el sistema de clasificación de patrones debe diseñarse partiendo de un conjunto de patrones de entrenamiento para los cuales no conocemos sus etiquetas de clase. Estas situaciones se presentan cuando no disponemos del conocimiento de un experto o bien cuando el etiquetado de cada muestra individual es impracticable. De todos modos, se pueden encontrar algoritmos o técnicas de aprendizaje automático que tienen la capacidad de aprender de ambas maneras, tanto supervisada como no supervisada.

En el campo de la detección de intrusión, se ha venido realizando un gran número investigaciones utilizando numerosas técnicas de aprendizaje, (Tsai*, et*

al., 2009). Aquí, la minería de datos y los métodos de aprendizaje automático se centran en el análisis de las propiedades de los patrones de auditoría en lugar de identificar el proceso que los genera. Normalmente se incluyen la extracción de reglas de asociación, clasificación y análisis de agrupamiento. Los métodos de clasificación han sido los métodos más investigados e incluyen arboles de decisiones, clasificadores Bayesianos, redes neuronales artificiales, clasificación del vecino más cercano (*k-nearest neighbour*), maquinas de soporte vectorial, algoritmos genéticos, etc. A continuación se detallan las técnicas más conocidas y las más aplicadas dentro del campo de los IDS.

3.3.1 Redes Neuronales Artificiales

Una Red Neuronal Artificial (*Artificial Neural Network, ANN*) es un modelo matemático inspirado en el comportamiento biológico de las neuronas y en la estructura del cerebro. Existen modelos muy diversos de redes neuronales en los cuales se siguen filosofías de diseño, reglas de aprendizaje y funciones de construcción de las respuestas muy distintas (Viñuela y León, 2004).

Cuando se trabaja con redes neuronales artificiales, se parte de un cierto modelo de neuronas y de una determinada arquitectura de red. Sin embargo, para que una red neuronal resulte operativa es necesario entrenarla, lo que constituye el modo de aprendizaje (Del Brío y Molina, 2006). Así pues, el aprendizaje se convierte en la parte más importante cuando se construye un sistema de neuronas artificiales. Los dos tipo básicos de aprendizaje son el supervisado y el no supervisado, pero pueden distinguirse muchos otros como por ejemplo el aprendiza hibrido y el reforzado.

Los tipos de redes neuronales más conocidos son las redes de base radial, los mapas autoorganizados de Kohonen, las redes de Hopfield y el Perceptrón Multicapa (MLP) (Viñuela y León, 2004), siendo este último el tipo de red aplicado en este proyecto. El MLP es una generalización del Perceptrón simple y surgió como consecuencia de las limitaciones de dicha arquitectura en lo referente al problema de la separabilidad no lineal. La arquitectura del MLP se caracteriza porque tiene sus neuronas agrupadas en capas de diferentes niveles. Así pues, cada una de las capas está formada por un conjunto de neuronas y se distinguen tres tipos de capas diferentes: la capa de entrada donde las neuronas se encargan únicamente de recibir las señales o patrones provenientes del exterior y propaga dicha señal a todas las neuronas de la siguiente capa. La capa oculta donde las neuronas realizan un procesamiento no lineal de los patrones recibidos, y por último la capa de salida que actúa como salida de la red, proporcionando al exterior la respuesta de la red para cada uno de los patrones de entrada. Finalmente, el MLP suele entrenarse mediante el algoritmo

denominado retropropagación de errores o BP (*BackPropagation*), o bien haciendo uso de alguna de sus variantes (Viñuela y León, 2004). La habilidad del MLP para aprender a partir de un conjunto de ejemplos, aproximar relaciones no lineales, filtrar ruidos en los datos, etc. hace que sea un modelo adecuado para abordar problemas reales. Tanto es así que el MLP es en la actualidad uno de las arquitecturas más utilizadas en la resolución de problemas, debido fundamentalmente a su capacidad como aproximador universal, así como a su fácil uso y aplicabilidad. Sin embargo, también es importante aclarar que aunque sea una de las redes más conocidas y utilizadas, esto no implica que siempre sea una de las más potentes y con mejores resultados en las diferentes áreas de aplicación.

Hay muchos problemas diferentes que pueden ser solucionados con una red neuronal artificial. Sin embargo las redes neuronales artificiales son comúnmente usadas para direccionar tipos de problema particulares, que incluyen problemas de clasificación, predicción, reconocimiento de patrones y optimización (Heaton, 2008).

En la detección de intrusión, se han realizado numerosos trabajos con redes neuronales artificiales (Beqiri, 2009) tratando de dar una alternativa a los sistemas expertos gracias a su flexibilidad y adaptación a los cambios naturales que se pueden dar en el entorno y, sobre todo, a la capacidad de detectar instancias de los ataques conocidos. El primer modelo de detección de intrusos basado en redes neuronales lo realizaron (Fox*, et al.*, 1990) como método para crear perfiles de comportamiento de usuarios. Debar y Dorizzi (1992) aplican el uso de la red neuronal para predecir el siguiente comando de una secuencia de comandos previos ejecutados por un usuario. Cansian *et al*., (1997) proponen el uso de redes neuronales para la identificación del comportamiento intrusivo en patrones de tráfico. Un IDS de red llamado INBOUNDS (*Integrated Network-Based Ohio University Network Detective Service*) donde un módulo de detección de anomalías basado en análisis estadístico se ha sustituido por otro que utiliza mapas auto-organizativos (Ramadas*, et al.*, 2003). Finalmente, enfoques IDS han sido aplicados recientemente para la detección de ataques de inyección SQL (Skaruz y Seredynski, 2007).

3.3.2 Árboles de Decisión y Sistemas de Reglas

De todos los métodos de aprendizaje, los sistema de aprendizaje basados en árboles de decisión son quizás el método más fácil de utilizar y entender (Orallo*, et al.*, 2004). Un árbol de decisión es un conjunto de condiciones organizadas en una estructura jerárquica, de tal manera que la decisión final a tomar se puede

determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta alguna de sus hojas.

Concretamente, los árboles se componen de dos tipos de nodos, los nodos hojas y los nodos de test. Los nodos hojas, se corresponden con un nodo final del árbol que contiene la información del grupo final en el que los casos son clasificados, mientras que los nodos de test, contienen una regla lógica que determina el subárbol a seleccionar según el valor que tome. En caso de ser cierto, se selecciona la rama de la izquierda y en caso de no cumplirse la condición se selecciona la rama de la derecha. De este modo, los casos se van disgregando según el valor de los nodos de test aplicando la estrategia de divide y vencerás.

Los árboles de decisión, permiten extraer el conocimiento en problemas en los que se dispone de una colección de casos, que contienen un conjunto de atributos que se consideran independientes y un atributo que se considera dependiente. Los atributos pueden ser de diversos tipos, pudiendo ser continuos o discretos, o bien nominales y ordinales. La meta es determinar el conjunto de atributos independientes que implican un determinado valor para el atributo dependiente.

Los árboles de decisión son una técnica que permite extraer de forma sencilla las reglas que permiten explicar las clasificaciones de los individuos. Las reglas se pueden expresar mediante antecedentes y consecuentes o bien se puede representar la información en forma de árbol lo que facilita la comprensión. Las últimas variantes de los algoritmos, permiten la inclusión de condiciones de decisión más variables, permitiendo incorporar atributos de tipo numérico. Sin embargo, el uso de atributos numéricos requiere discretizar los valores para poder aplicar estos procedimientos cuando el número de atributos es elevado. Si no se lleva a cabo la discretización de los valores, la selección del punto de corte para las reglas de decisión se convierte en una etapa demasiado pesada para el procesamiento.

Por otro lado, los sistemas de reglas son una generalización de los árboles de decisiones en el que no se exige exclusión ni exhaustividad en las condiciones de las reglas, es decir podría aplicarse más de una regla o ninguna. La representación en formas de reglas suele ser, en general, más sucinta que la de los árboles, ya que permite englobar condiciones y permite el uso de reglas por defecto (Orallo, *et al.*, 2004).

Basándose en diferentes particiones, en un criterio de partición y otras extensiones, han aparecido números algoritmos o sistemas de aprendizaje de árboles de decisión, por ejemplo los más populares CART (*Classification and Regression Trees*) (Bittencourt y Clarke, 2003), ID3 (Quinlan, 1986), C4.5 (Quinlan, 1986).

Los arboles de decisión y los sistemas de reglas presentan ventajas como su aplicabilidad a varias tareas de la minería de datos como la clasificación, regresión, agrupamiento y estimación de probabilidades. Son fáciles de usar y son tolerantes al ruido, a atributos no significativos y a valores faltantes. Además de que pueden tratar con atributos numéricos y nominales.

En la detección de intrusión, una propuesta de árbol de decisión fue propuesta como método de aprendizaje de firmas de ataques. Dicho clasificador aprendía firmas de intrusiones en su fase de entrenamiento, para después clasificar en diferentes estados las actividades de sistemas informáticos, y predecir la posibilidad de que ocurriese un ataque (Ye y Li., 2000). Otro enfoque para la generación de firmas de ataque de manera automática para su uso en la detección de uso indebido en redes fue propuesto por (Han*, et al.*, 2002). Su objetivo era crear una herramienta de minería de datos que ayudara a expertos en el descubrimiento de de firmas o patrones de ataque. Una variante del algoritmo ID3 es propuesto por (Krüegel*, et al.*, 2002), utilizando arboles de decisiones como método de optimización de IDS basados en análisis de firma. Una implementación basada en una combinación de redes neuronales con el algoritmo C4.5 para la detección de usos indebidos fue presentada por (Pan*, et al.*, 2003). Finalmente, en la detección de ataques de inyección SQL se han propuestos algunos trabajos (Bertino*, et al.*, 2007), (García*, et al.*, 2006).

3.3.3 Métodos Bayesianos

Los métodos bayesianos a diferencia de otras técnicas de la minería de datos pueden trabajar con incertidumbre, ya que una de sus principales características es el uso explicito de la teoría de la probabilidad para cuantificar incertidumbre. La teoría de la probabilidad y los métodos bayesianos son unas de las técnicas que más se han utilizado en problemas de Inteligencia Artificial y, por tanto, de aprendizaje automático y minería de datos (Orallo*, et al.*, 2004).

Dentro de los métodos bayesianos encontramos la red bayesiana, que es un modelo probabilístico multivariante que relaciona a un conjunto de variables aleatorias mediante una grafo dirigido acíclico, que permite inferencia bayesiana para la estimación de probabilidades de variables no conocidas a partir de variables conocidas. Las redes bayesianas constan de los siguientes elementos:

- Un conjunto de nodos, uno por cada variable aleatoria.
- Un conjunto de arcos dirigidos que conectan los nodos.

- Cada nodo contiene la distribución de probabilidad condicional que lo relaciona con el nodo padre.

Se fundamentan en teorema de Bayes (Duda y Hart, 1973), que permite calcular la probabilidad de un suceso a priori sabiendo que ha ocurrido a posteriori el suceso.

Otro de los métodos y sin duda alguna uno de los modelos más simple de clasificación con redes bayesianas es el clasificador *Naïve Bayes*. El fundamento principal del clasificador *Naive Bayes* es la suposición de que todos los atributos son independientes conocido el valor de la variable clase (Duda y Hart, 1973). La hipótesis de independencia asumida por el clasificador da lugar a un modelo gráfico probabilístico en el que existe un único nodo raíz (la clase), y en la que todos los atributos son nodos hoja que tienen como único padre a la variable clase. Es uno de los clasificadores más utilizados y diversos estudios (Michie*, et al.*, 1994) demuestran que sus resultados son competitivos con otras técnicas (redes neuronales, árboles de decisiones, etc.) en muchos problemas y que incluso las superan en algunos otros.

Los clasificadores bayesianos han sido aplicados en la detección de intrusiones. Dentro de las alternativas propuestas está el desarrollo de un módulo para el IDS EMERALD llamado eBayes TCP que utilizaba tecnología de redes Bayesianas para analizar, lo que ellos denominan, "explosiones de tráfico". Las categorías de ataques se representan como hipótesis de modelos, las cuales se van reforzando de forma adaptativa (Valdes y Skinner, 2000). Daniel Barbará *et al*., (2001) también hacen uso de la teoría Bayesiana para su sistema ADAM (*Audit Data Analysis and Mining*). En dicho trabajo, proponen el uso de estimadores pseudo-Bayes para afinar la capacidad detectar anomalías, reduciendo a su vez el número de falsas alarmas. Un clasificador *Naïve Bayes* para detectar intrusiones sobre eventos de red fue propuesto por (Schuba*, et al.*, 1997). Kruegel & Vigna. (2003) utilizaron redes Bayesianas para clasificar eventos basados en las salidas de los diferentes modelos para la detección de anomalías y en información adicional extraída del mismo entorno. Finalmente, un reciente trabajo propuesto se basa en la utilización de una red bayesiana grande para la detección de eventos de intrusiones en redes complejas (Tuba y Bulatovic, 2009).

3.3.4 Lógica Difusa

La denominada lógica borroza (*fuzzy logic*) (Zadeh, 1965), permite tratar información imprecisa en términos de conjuntos borrosos difusos. Permite modelar conocimiento impreciso y cuantitativo así como transmitir, manejar y

soportar, en una extensión razonable, el razonamiento humano de una forma natural (Del Brío y Molina, 2006).

Los sistemas basados en reglas que utilizan conjuntos difusos para describir los valores de sus variables se denomina sistemas basados en reglas difusas. Se basa en reglas heurísticas de la forma *SI* (antecedente) *ENTONCES* (consecuente), donde el antecedente y el consecuente son también conjuntos difusos, ya sea puros o resultado de operar con ello. Los conjuntos borrosos pueden ser considerados como una generalización de los conjuntos clásicos (Klir y Yuan, 1995): la teoría clásica de conjuntos sólo contempla la pertenencia o no pertenencia de un elemento a un conjunto (0 ó 1), sin embargo la teoría de conjuntos difusos contempla la pertenencia parcial de un elemento a un conjunto, es decir, cada elemento presenta un grado de pertenencia a un conjunto difuso que puede tomar cualquier valor entre 0 y 1. Este grado de pertenencia se define mediante la función característica asociada al conjunto difuso: para cada valor que pueda tomar un elemento o variable de entrada *x* la función característica $\mu_A(X)$ proporciona el grado de pertenencia de este valor de *x* al conjunto difuso *A*.

Los sistemas basados en lógica difusa imitan la forma en que toman decisiones los humanos, con la ventaja de ser mucho más rápidos. Estos sistemas son generalmente robustos y tolerantes a imprecisiones y ruidos en los datos de entrada. La verdadera aplicabilidad de la lógica difusa se da cuando la complejidad del proceso en cuestión es muy alta y no existen modelos matemáticos precisos, para procesos altamente no lineales y cuando se envuelven definiciones y conocimiento no estrictamente definido (impreciso o subjetivo). La lógica difusa se utiliza para la resolución de una variedad de problemas, principalmente los relacionados con control de procesos industriales complejos y sistemas de decisión en general, la resolución de la compresión de datos.

La lógica borrosa se ha convertido en un área de investigación en la detección de intrusos por dos razones importantes. Por un lado, están involucradas una gran cantidad de características cuantitativas, y por el otro, la seguridad en sí misma incluye la incertidumbre, es un hecho borroso (Bridges*, et al.*, 2000). Dada una característica cuantitativa, se puede usar un intervalo para indicar un valor normal. El primer trabajo sobre el uso de la aplicación de la lógica borrosa se le debe a Lin (1994). Sin embargo, no fue hasta principios de esta década cuando realmente se plantean numerosos enfoques IDS basdos en lógica borrosa. Un enfoque IDS llamado FIRE (*Fuzzy Intrusion Recognition Engine*) utiliza agentes borrosos que monitorizan una red en busca de signos de intrusión a bajo nivel. La relación entre agentes de bajo nivel y los agentes de mayor nivel se modela mediante Mapas Cognitivos Borrosos (*Fuzzy Cognitive Maps*) que sirven a modo de base de conocimiento para diferentes tipos de intrusión (Dickerson y Dickerson, 2000), (Xin*, et al.*, 2003). Zhang Jiang *et al*., (2003) hacen uso de la teoría borrosa para el motor de razonamiento y respuesta

de los IDS. Con el enfoque se demostró que la técnica supera los IDS basados en sistemas expertos tradicionales al aumentar la velocidad de detección y disminuir el costo acumulado de la detección de intrusos en relación a las respuestas no estáticas. Otra propuesta, esta vez para definir el comportamiento normal y anómalo de una red fue propuesto por (Guan*, et al.*, 2004). Un enfoque reciente se basa en una combinación de tres técnicas de aprendizaje automático, agrupamiento por medio de *K-Means*, lógica borrosa y redes neuronales. Mediante el uso de un modelo de lógica borrosa se genera las reglas a partir de las características percibidas de los clústeres normales y anormales (Kumar y NandaMohan, 2008). Finalmente, una propuesta reciente basada en una infraestructura de detección de intrusión es propuesta por (Tajbakhsh*, et al.*, 2009). Los algoritmos usados en el motor de clasificación usan reglas de asociación borrosa para construir los clasificadores.

3.3.5 Algoritmos Genéticos

Los algoritmos genéticos (*AGs*) propuestos por Holland (Holland, 1975), son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están inspirados en el modelo de la evolución genética de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acuerdo con los principios de selección natural y la supervivencia de los más aptos (postulados de Darwin). Así pues, por imitación de este proceso, los algoritmos genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas.

Como se resume en (Moujahid*, et al.*, 2004), los algoritmos genéticos usan una analogía directa con el comportamiento natural. Trabajan con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna un valor o puntuación, relacionado con la bondad de dicha solución. En la naturaleza esto equivaldría al grado de efectividad de un organismo para competir por unos determinados recursos. Cuanto mayor sea la adaptación de un individuo al problema, mayor sería la probabilidad de que el mismo sea seleccionado para reproducirse, cruzando su material genético con otro individuo seleccionado de igual forma. Este cruce producirá nuevos individuos (descendientes de los anteriores) los cuales comparten algunas de las características de sus padres. Cuanto menor sea la adaptación de un individuo, menor sería la probabilidad de que dicho individuo sea seleccionado para la reproducción, y por tanto de que su material genético se propague en sucesivas generaciones.

Así pues, de esta manera se produce una nueva población de posibles soluciones, la cual reemplaza a la anterior y verifica la propiedad interesante que contiene una mayor proporción de buenas características en comparación con la población anterior. Así, a lo largo de las generaciones las buenas características se propagan a través de la población. Favoreciendo el cruce de los individuos mejor adaptados, van siendo exploradas las tareas más prometedoras del espacio de búsqueda. Si el algoritmo genético ha sido bien diseñado, la población convergerá hacia una solución óptima del problema.

Los algoritmos genéticos gozan de una buena aceptación al ser una técnica robusta que puede ser aplicada con éxito en una gran variedad de problemas provenientes de diferentes áreas, incluyendo aquellos en los que otros métodos encuentran dificultades. El gran campo de aplicación de los algoritmos genéticos se relaciona con aquellos problemas para los cuales no existen técnicas especializadas. Incluso en el caso en que dichas técnicas existan, y funcionen bien, pueden efectuarse mejoras de las mismas hibridándolas con los algoritmos genéticos. En la detección de intrusión, los algoritmos genéticos son usados con el fin de mejorar la eficiencia seleccionando subconjuntos de características para reducir el número de características observadas manteniendo, o incluso mejorando, la precisión del aprendizaje. A diferencia del resto de las técnicas de aprendizaje automático, los algoritmos genéticos son recientes en el campo de los IDS. Uno de los proyectos propuestos es GASSATA (*Genetic Algorithm as an Alternative Tool for Security Audit Trail Analysis*) (Me, 1998), el cual utiliza un algoritmo genético para buscar la combinación de los ataques conocidos que mejor se correspondan con el evento (o registro de auditoría) observado. Helmer *et al*., (1999) utilizaron algoritmos genéticos como método para seleccionar subconjuntos de características a partir de vectores de características que describían las llamadas al sistema ejecutadas por procesos con privilegios. Dicha selección permitía reducir significativamente el número de características necesarias para la detección sin que ello afectara a la precisión. *Intelligent Intrusion Detection System* (IIDS) (Li, 2004) es un proyecto IDS que combina detección basada en anomalía y uso indebido, además de trabajar como IDS a nivel host y de red. A nivel del IDS, se clasifica un conjunto de conexiones de red entre normal o intrusiva de forma manual. El algoritmo genético se inicia con un pequeño conjunto de reglas generadas aleatoriamente, y dichas reglas evolucionan hasta generar un conjunto de datos mayor que contiene las nuevas reglas del IDS. Finalmente, un enfoque IDS utilizando el método de aprendizaje basado en genética difusa fue propuesto por (Abadeh*, et al.*, 2007). Mediante la estrategia se logró producir más reglas difusas las cuales eran más efectivas para la detección de intrusión.

3.3.6 Máquina de Soporte Vectorial

Las maquinas de soporte vectorial (*Support Vector Machine, SVM*) (Vapnik, 1995), fueron desarrolladas para el problema de clasificación pero luego evolucionaron a la forma actual de SVM para regresión Vaptnik (Vapnik*, et al.*, 1996). SVM está ganando gran popularidad como herramienta para la identificación de sistemas no lineales, esto debido principalmente a que SVM está basado en el principio de minimización del riesgo estructural (SRM, *Structural Risk Minimization*).

El objetivo final de un algoritmo SVM es encontrar el hiperplano óptimo que separe en dos grupos el contenido del vector, tomando como punto de separación una variable predictora. De esta forma, los puntos del vector que se encuentren situados dentro de los márgenes de una categoría de la variable estarán a un lado del hiperplano y los casos que se encuentren en la otra categoría estarán al otro lado.

Los modelos basados en SVMs están estrechamente relacionados con las redes neuronales. Usando una función *kernel*, proporcionan un método de entrenamiento alternativo para clasificadores polinomiales, redes de base radial y PERCEPTRÓN multicapa.

Las SVM presentan ventajas comparadas con otras técnicas como es el caso de (Orallo*, et al.*, 2004):

- Un entrenamiento relativamente fácil
- No hay óptimo local a diferencia de las redes neuronales.
- Se escalan relativamente bien para datos en espacios dimensionales altos.
- El compromiso entre la complejidad del clasificador y el error puede ser controlado explícitamente.
- Datos no tradicionales como cadenas de caracteres y árboles pueden ser usados como entrada a la SVM, en vez de vectores de características.

Los campos donde las SVM han sido aplicadas con éxito incluyen, entre otros, la visión por computador, la bioinformática, la recuperación de información, el procesamiento del lenguaje natural y el análisis de serie temporal (Orallo*, et al.*, 2004). Dentro del campo de la detección de intrusiones, las SVM permiten manejar larga dimensionalidad de los datos y ejecutar clasificación multi-clase. Algunos de los enfoques propuestos está (Ambwani, 2003), el cual utiliza clasificadores SVM múltiples, usando el método uno-contra-uno, para la detección de anomalías y también de uso indebido, identificando de una forma

precisa los ataques según su tipo. Mukkamala *et al.,* (2002) planteó utilizar cinco SVM tradicionales, uno para identificar tráfico normal, y el resto para identificar cada uno de los cuatro tipos de ataques (DoS, R2L, U2R, *Probing*). Luego de una comparación con redes neuronales se llegó a la conclusión que las SVM demostraban mejor desempeño. Un enfoque reciente en la detección de intrusiones ha sido propuesto utilizando SVM multi-clase (Xin WEI y qing WU, 2008). Los resultados mostraron que la solución propuesta puede mejorar la exactitud en la clasificación y reducir el tiempo de entrenamiento.

3.3.7 Método de Agrupamiento (*Clustering*)

El agrupamiento consiste en encontrar grupos entre un conjunto de individuos. Permite descubrir asociaciones y estructuras en los datos que no son evidentes a priori pero que pueden ser útiles una vez que se han encontrado (Orallo*, et al.*, 2004). Los resultados de un Análisis de *Clusters* pueden contribuir a la definición formal de un esquema de clasificación tal como una taxonomía para un conjunto de objetos, a sugerir modelos estadísticos para describir poblaciones, a asignar nuevos individuos a las clases para diagnóstico e identificación, etc.

Los dos tipos fundamentales de métodos de clasificación son el jerárquico y no jerárquico. En los métodos de clasificación jerárquicos, la clasificación resultante tiene un número creciente de clases anidadas mientras que en el segundo las clases no son anidadas. Además, los métodos pueden dividirse en aglomerativos y divisivos. En los primeros se parte de tantas clases como objetos tengamos que clasificar y en pasos sucesivos vamos obteniendo clases de objetos similares, mientras que en los segundos se parte de una única clase formada por todos los objetos que se va dividiendo en clases sucesivamente

El concepto de distancia puede jugar un papel crucial ya que la manera de formalizar el concepto de similitud es a través de métricas o medidas de distancia. Si se quiere saber la similitud entre dos instancias o individuos, es necesario elegir una función de distancia y calcular con ella la distancia entre los dos individuos. La distancia *Euclidea* o clásica es las más conocida sin embargo existen otras que cumplen los requisitos de una función de distancia (métrica) y que pueden funcionar mejor en ciertos contextos. (Orallo*, et al.*, 2004).

Los métodos de agrupamiento han sido aplicados en la detección de intrusión usando diferentes métodos tales como *K-Means*, *Fuzzy C-Means* (Portnoy*, et al.*, 2001), (Shah*, et al.*, 2003). En la detección de ataques de inyección SQL encontramos un enfoque propuesto por (Bockermann*, et al.*, 2009). Debido a que las técnicas de agrupamiento requieren una medida de

distancia numérica, las observaciones requieren ser numérica, lo que a nivel de los sistemas de detección de intrusión podría ser una importante limitación.

El aprendizaje automático se vislumbra como un campo clave de investigación para solucionar muchos de los problemas actuales en la detección de intrusiones. El aprendizaje automático proporciona algoritmos robustos que permiten que un sistema tenga la capacidad de aprender a partir de ejemplos. Tomando en cuenta el potencial que ofrece el campo del aprendizaje automático es posible pensar en la incorporación de técnicas clasificación y predicción novedosas para determinar la fiabilidad de las peticiones de usuario. Un mecanismo de detección que utilice árboles de decisión para detectar ataques simples y redes neuronales para los ataques que requieren una técnica de clasificación más robusta parece adecuado para afrontar el problema de los ataques de inyección.

3.4 Tipo de Agente CBR-BDI

La detección de amenazas a nivel de la capa de aplicación supone enfrentarse a un entorno extremadamente dinámico, donde continuamente nuevas vulnerabilidades van siendo descubiertas y nuevas amenazas van apareciendo y evolucionando. Así pues, para afrontar este escenario se requiere flexibilidad y capacidad de adaptación para hacer frente a los cambios en el comportamiento de los ataques y los usuarios maliciosos. Los agentes y sistemas multi-agente, especialmente aquellos basados en una estructura deliberativa han ganado relevancia a la hora de llevar a cabo el desarrollo de aplicaciones en entornos dinámicos y flexibles, tales como la Web (Hendler, 2006), interfaces personalizados de usuario (Middleton, 2001), oceanografía (Bajo y Corchado, 2005), (Corchado*, et al.*, 2007), sistemas de control (Jennings y Bussmann, 2003), entornos de robótica (Bajo*, et al.*, 2008), (Busquets*, et al.*, 2003), (Sierra*, et al.*, 2001) y en la detección de intrusiones (Herrero*, et al.*, 2009), (Orfila*, et al.*, 2008).

Los agentes vienen caracterizados a través de sus capacidades, tales como autonomía, reactividad, pro-actividad, habilidades sociales, razonamiento, aprendizaje y movilidad entre otras. No obstante, estas capacidades iniciales de los agentes puedan que no sean suficientes para resolver tareas complejas en la detección de intrusiones. Como las capacidades de los agentes se pueden modelar de distintas formas y con diferentes metodologías (Wooldridge y Jennings, 1995), una de las posibilidades es la de incrementar las capacidades de los agentes utilizando sistemas de razonamiento basado en casos (CBR).

Una parte fundamental de esta propuesta es la utilización de agentes deliberativos BDI que hacen uso de sistemas CBR, también conocidos como

agentes CBR-BDI (Pinzón, *et al.*, 2009c), (Pinzón, *et al.*, 2008b), (Corchado y Laza, 2003), (Corchado, *et al.*, 2004), (Bajo, *et al.*, 2006b). Utilizando el razonamiento basado en casos (*Case Base Reasoning*, CBR) (Laza, *et al.*, 2003), los problemas se resuelven mediante la adaptación de soluciones que fueron propuestas previamente para resolver problemas similares (Riesbeck y Schank, 1989).

A continuación se presenta una formulación que permite relacionar los conceptos que definen la estructura interna y el comportamiento de un agente BDI con los conceptos manejados en sistemas de razonamiento basado en casos.

La generación del modelo CBR-BDI permite integrar dentro de la arquitectura deliberativa BDI el motor de razonamiento de un CBR, para ello, es necesario establecer una relación entre la información usada por el CBR y la arquitectura BDI de modo que ambos puedan interaccionar. La estructura de un sistema CBR se ha diseñado entorno al concepto de caso. Cada uno de los casos va a estar formado por una terna formada por la descripción del problema, la solución y el resultado obtenido (Laza, *et al.*, 2003). Esta descomposición de un CBR se puede ver en la Figura 3.7.

CBR	BDI
Caso: Problema, solución, resultado **Problema**: estado_inicial **Solución**: {acción, [estado_intermedio]}* **Resultado**: estado_final	**Creencias:** {estado_final}|{estado acción, estado} **Intenciones:** {creencias}+ **Deseos:** {estado_final}
Sintaxis	
{}: Secuencia []: Opcional *: 0 a n repeticiones +: 1 a n repeticiones |: disyunción BDI	

Figura 3.7. Caso CBR-BDI (Aamodt y Plaza, 1994)

Los componentes de la estructura del caso CBR-BDI se presentan en la Figura 3.7. Para el caso del CBR, se muestran los componentes: el problema define la situación del entorno en un momento dado, la solución es el conjunto de estados del entorno como consecuencia de las acciones que son llevadas a cabo, y el resultado muestra la situación del entorno una vez que el problema ha sido resuelto. En el caso BDI, se definen las creencias, los deseos y las intenciones de un agente. Los cambios entre estados después de llevar a cabo una acción se considera una creencia (los agentes recuerdan las acciones que son llevadas a cabo bajo una determinada situación y el resultado obtenido). Los objetivos que un agente quiere alcanzar a partir de un estado en un ambiente conocido son lo que se conoce como deseos. Las intenciones son la serie de planes que debe seguir para cumplir los objetivos, no son más que una serie creencias que han

sido ordenadas. Así, en función de la información mostrada, se puede ver que existe una relación entre la descripción del caso del CBR y las creencias del modelo BDI.

En resumen, la utilización de agentes CBR-BDI es un avance significativo en el estado del arte de la detección de intrusiones. Durante el continuo proceso de revisión del estado del arte, no se ha podido constatar, enfoque alguno bajo la premisa de la detección que utilice esta técnica. Los agentes CBR-BDI poseen la habilidad de aprender, razonar y adaptarse continuamente a los cambios de comportamientos en los ataques. Estas capacidades le aseguran al mecanismo de detección de la arquitectura una capacidad incremental de aprendizaje y adaptación, en el tiempo, sin importar la velocidad con que los ataques puedan evolucionar.

3. 5 Conclusiones

En este capítulo se han presentado los aspectos más importantes de las tecnologías y técnicas aplicadas en la elaboración de este trabajo de tesis doctoral.

Los sistemas multi-agente resultan adecuados para resolver problemas en aquellos entornos caracterizados por la dinámica y el manejo descentralizado de los datos y las operaciones. La detección de intrusiones en los entornos de las aplicaciones distribuidas supone un escenario adecuado para la implementación de sistemas multi-agente. Sin embargo, el problema de detección no se resuelve aplicando los agentes simplemente y confiando en sus capacidades. Los nuevos tipos de amenazas presentan como principal característica la capacidad de evolucionar rápidamente o camuflarse para sortear las medidas de seguridad instaladas. En este sentido se requiere mayores capacidades de adaptabilidad y aprendizaje para hacer frente a las nuevas amenazas. Para alcanzar este objetivo, la propuesta que se presenta, plantea el uso de agentes deliberativos BDI utilizando sistemas CBR. Los sistemas CBR se fundamentan en la idea que los problemas similares tienen soluciones similares. Con lo cual, esta estrategia encaja de manera especial en la mecánica para la detección de intrusiones, permitiendo dotar al sistema de una capacidad de aprendizaje incremental a través de la experiencia. Este tipo de agente se conoce como agentes CBR-BDI.

Finalmente, queda el problema de resolver la identificación de actividades maliciosas. La tarea más importante en el campo de la detección de intrusión es lograr filtrar las actividades maliciosas de aquellas que no lo son. Esta tarea ha resultado ser el mayor desafío en el ámbito de la detección de intrusiones. En este capítulo se ha realizado una revisión de las principales técnicas aplicadas en

la detección de intrusiones. Una de las áreas que ha evolucionado rápidamente y promete soluciones eficientes en la detección de intrusiones son las técnicas de aprendizaje automático. Mediante la aplicación de estas técnicas, muchas de las limitaciones de los enfoques IDS tradicionales pueden ser superadas. En nuestro caso, se han revisado varías técnicas y se han seleccionado las redes neuronales y los árboles de decisión, ya que ambas técnicas han demostrado su eficiencia a la hora de descubrir patrones maliciosos que suponen un riesgo de seguridad, tal y como se puede apreciar en los casos de estudio presentados en el Capítulo 5. Sin embargo, técnicas como los algoritmos genéticos, la máquina de soporte vectorial y las técnicas de agrupamiento (*clustering*) presentan la desventaja de requerir una alta carga computacional. Por ello, pese a que suponen estrategias robustas e innovadoras en el campo de la detección de intrusiones, su implementación depende en gran medida del entorno de aplicación, y resulta poco eficiente en entornos que demandan una solución en tiempo real como lo es la detección de intrusiones en entornos distribuidos.

El resultado de la integración de las tecnologías y técnicas presentadas en este capítulo da la posibilidad de construir un enfoque innovador a nivel de la tecnología de los IDS, y más específicamente una solución IDS eficaz para hacer frente a las nuevas amenazas en los entornos de las aplicaciones distribuidas. En el siguiente capítulo se presentará de forma detallada los componentes de la arquitectura AIDeMaS y como se ha logrado integrar las distintas tecnologías aquí discutidas.

Capítulo 4

Arquitectura AIDeMaS

Introducción

En la actualidad los usuarios tienen acceso a tecnologías que les permiten obtener información de una manera ubicua y en tiempo real. La aparición de nuevos dispositivos con conexiones inalámbricas permiten a los usuarios moverse de un lado a otro llevando consigo las aplicaciones que años atrás estaban instaladas en un equipo de escritorio. Los dispositivos ligeros habitualmente utilizan conexiones remotas a servidores de datos para obtener la información que el usuario solicita. Esta apertura en la comunicación y la urgente necesidad de alcanzar una verdadera interoperabilidad entre entornos de aplicaciones heterogéneas, junto con las grandes ventajas y capacidades que ofrece este nuevo modelo de interacción, han supuesto por otra parte la necesidad de investigar con respecto a la seguridad de las aplicaciones y los datos.

Los problemas de seguridad de las aplicaciones emergentes y los datos son causados generalmente debido al acceso continuo por distintos tipos de usuarios. Además, las nuevas aplicaciones de la Web manejan información distribuida en distintos puntos geográficos haciendo que la protección de los datos y la seguridad de las aplicaciones sean más complicadas y, en muchas ocasiones, imposible de garantizar plenamente. Finalmente, hay que añadir los problemas de seguridad relativos a los canales de comunicación por donde viaja la información, como es el caso de Internet. Tomando en cuenta estos y muchos otros riesgos, la seguridad informática se ha convertido en un área de investigación en continua evolución para hacer frente a las constantes amenazas en los entornos de las aplicaciones distribuidas.

En este trabajo se presenta una arquitectura de seguridad para entornos distribuidos, AIDeMaS: Sistema Multi-Agente para la Detección de Intrusiones de forma Adaptativa (en Inglés, *Adaptive Intrusion Detection Multi-Agent System*), diseñada para detectar y bloquear intrusiones que tienen lugar en la capa de aplicación de los entornos de aplicaciones distribuidas. Inicialmente la arquitectura ha sido diseñada para abordar intrusiones con características similares a los ataques de inyección SQL y ataques XDoS. Sin embargo, AIDeMaS se puede extender fácilmente para abordar otros tipos de intrusiones.

AIDeMaS está basada en una arquitectura multi-agente que utiliza un diseño jerárquico en capas. En cada una de las capas existen distintos tipos de agentes especializados en el desarrollo de las tareas específicas orientadas a garantizar la seguridad. Adicionalmente, el corazón de la arquitectura consiste en un mecanismo de clasificación en dos etapas llevado a cabo por agentes CBR-BDI con capacidades avanzadas. Estos agentes están dotados de una novedosa estructura interna que los dota de las habilidades y el conocimiento necesarios para identificar intrusiones en las peticiones de usuarios. En resumen, la arquitectura AIDeMaS está construida sobre la base de la tecnología multi-agente, técnicas y algoritmos del campo del aprendizaje automático y el paradigma CBR.

En este capítulo se describe en detalle cada uno de los componentes que componen la arquitectura. En la sección 4.1 se presenta la arquitectura desarrollada, en la sección 4.2 se describe el mecanismo de clasificación y en la sección 4.3 se presentan las conclusiones obtenidas.

4.1 Arquitectura Propuesta

La arquitectura AIDeMaS se ha planteado con el objetivo de proponer un nuevo enfoque, que supere las limitaciones de los mecanismos de seguridad existentes utilizados en la detección de intrusiones en los entornos de aplicaciones Web. Para ello, AIDeMaS incorpora capacidades de aprendizaje y adaptación que le permite evolucionar frente a los cambios de comportamientos de los usuarios maliciosos y las intrusiones. Además, AIDeMaS provee capacidad de computación distribuida para mejorar el rendimiento en la ejecución de las tareas. Finalmente, AIDeMaS utiliza un diseño jerárquico para modelar claramente la estrategia de detección, mejorar la distribución de las tareas y facilitar una mayor capacidad de recuperación de errores. Para alcanzar este objetivo, AIDeMaS integra un conjunto de tecnologías y técnicas modernas del campo de la seguridad y la Inteligencia Artificial.

En esta sección se describen los distintos componentes de la arquitectura AIDeMaS y la forma como los mismos interactúan. Sin embargo, antes de entrar en la descripción de los componentes de la arquitectura, es conveniente analizar el concepto de jerarquía y las ventajas que aporta utilizar una estructura organizada jerárquicamente como se plantea en esta propuesta.

4.1.1 Enfoque Jerárquico Distribuido

El diseño jerárquico se presenta como una solución para organizar y controlar grandes y complejas organizaciones o sistemas. Según (SchneeweiB, 1995), (Varaiya, 2000), existen razones importantes para considerar una estructura jerárquica:

- Entender de manera profunda la complejidad y comportamiento del sistema.
- Reducir la complejidad de la comunicación y computación.
- Alcanzar un esquema modular y adaptable a los cambios, así como también una robustez y escalabilidad.

Desde el punto de vista de la arquitectura AIDeMaS, la estructura jerárquica ofrece ventajas importantes que han sido consideradas. La estructura jerárquica facilita organizar las distintas tareas derivadas del proceso de detección de intrusión de acuerdo a un conjunto de roles definidos. Cada una de las 4 capas de la arquitectura AIDeMaS tiene asignados roles que a su vez encierran un conjunto de tareas. La forma como se ha definido cada rol en cada capa viene establecido por la estrategia global que se ha diseñado para lograr una efectividad en la detección sin impactar significativamente en el rendimiento de la aplicación. Al plantear una arquitectura multi-agente como estrategia para la detección de intrusiones conviene plantear una estructura organizacional para definir los roles de los agentes, entender el comportamiento de los agentes, y establecer la forma como se comunican los agentes. En la Figura 4.1 se presenta el diseño de la estructura jerárquica de la arquitectura AIDeMaS y la función asignada a cada capa de la arquitectura. Se han definido 4 funciones globales que corresponden a las 4 capas de la arquitectura AIDeMaS:

- Función Monitorización y Captura: Encierra las tareas de monitorización del tráfico para identificar y capturar las peticiones de usuarios dirigidas a las aplicaciones.
- Función de Clasificación Ligera: Encierra las tareas correspondientes a la ejecución de un proceso rápido para detectar anomalías en las peticiones de usuarios.
- Función de Clasificación Pesada: Encierra las tareas correspondientes a la ejecución de un proceso exhaustivo para detectar anomalías en las peticiones de usuarios.
- Función Administración: Encierra las tareas relacionadas a la administración y control de la arquitectura así como también a la gestión de alertas y toma de decisiones frente a situaciones de amenazas.

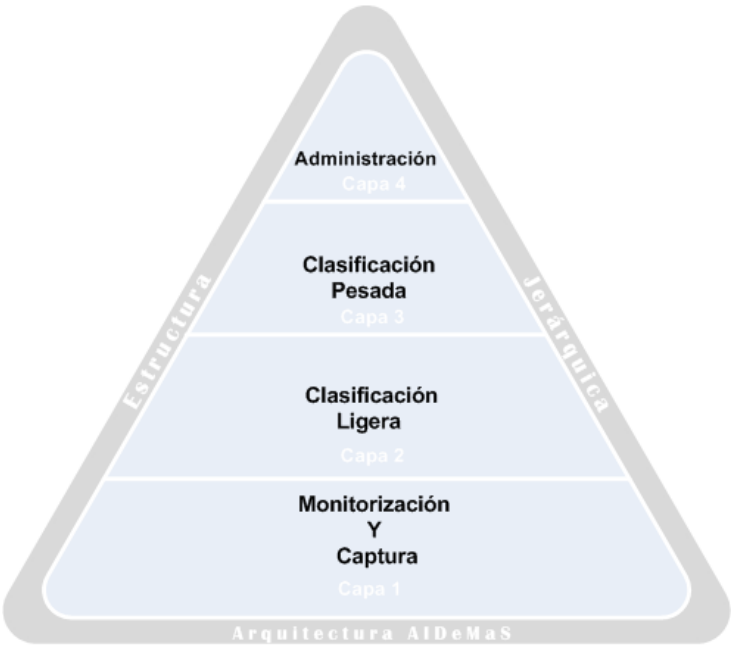

Figura 4.1. Diseño jerárquico de la arquitectura AIDeMaS y funciones definidas

La estructura jerárquica de la arquitectura AIDeMaS facilita el control de la comunicación de los agentes asignados dentro de las capas durante la ejecución de las tareas, y la comunicación entre capas durante el intercambio de información a nivel de las capas. Otra ventaja importante es la escalabilidad de la arquitectura al tener la capacidad de gestionar el conjunto de agentes necesarios en los casos donde se presente una carga de trabajo significativa. Mediante la estructura en capas se puede determinar fácilmente donde se está originando la mayor carga computacional para incorporar nuevos agentes. Finalmente, la estructura jerárquica permite estudiar la estrategia de detección como un mecanismo de clasificación por etapas, lo que facilita entender cómo se lleva el proceso de detección de intrusiones paso a paso desde el nivel más inferior de la arquitectura hasta el nivel superior. En conclusión, la estructura jerárquica resulta conveniente desde el punto de vista del diseño de la estrategia de detección y como recurso para simplificar la complejidad del propio diseño de la arquitectura.

Con respecto a la capacidad distribuida de la arquitectura AIDeMaS, esta viene dada por la propia naturaleza de movilidad con que cuentan los agentes. La capacidad de distribuir los agentes está estrechamente relacionada con la disponibilidad de los recursos computacionales (memoria, CPU, Red). Disponer de recursos computacionales garantiza la capacidad de mover los agentes a los

nodos disponibles, y llevar a cargo las operaciones de cómputo para resolver las tareas asignadas a los agentes correspondientes.

En la arquitectura AIDeMaS se plantea la distribución de funciones en las diferentes capas como se presenta en la Figura 4.2.

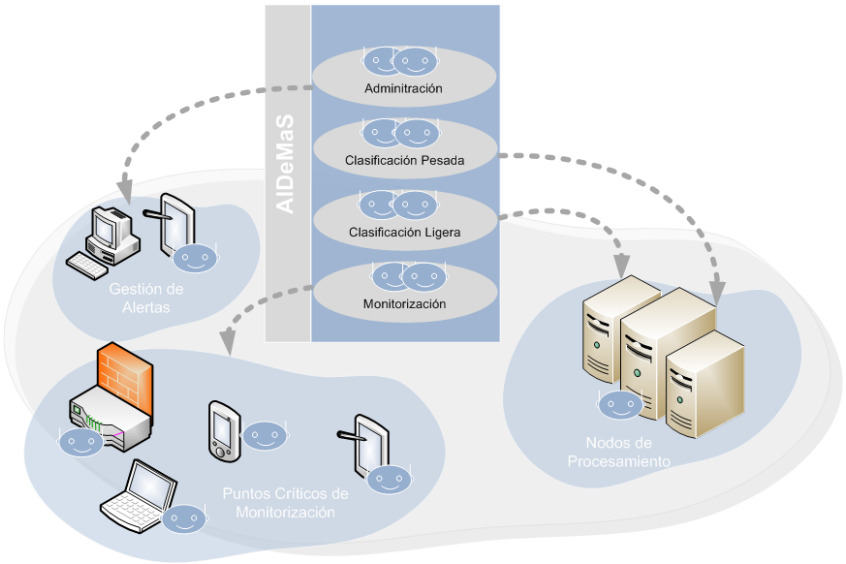

Figura 4.2. Diseño distribuido de la arquitectura AIDeMaS

- Capa 1: La monitorización y captura del tráfico se plantea para puntos críticos, ya sea a nivel de la red o desde nodos (dispositivos móviles) con capacidad de albergar agentes, y que dispongan de interfaces de usuario para lanzar peticiones de usuarios a las aplicaciones.

- Capa 2: Las tareas de clasificación ligera se plantean para ser distribuidas a nodos específicos para llevar a cabo está fase de clasificación. Esto permitirá un alto rendimiento en el procesamiento de las peticiones de usuario.

- Capa 3: Las tareas de la clasificación pesada igualmente se plantean para ser distribuidas a nodos específicos. En esta capa se llevan a cabo tareas que demandan una significativa cantidad de recursos de memoria y ciclos de CPU. Distribuyendo estas tareas a agentes situados en nodos específicos se logra alcanzar una mejora significativa en el procesamiento de las peticiones de usuario en esta etapa del mecanismo de clasificación.

- Capa 4: La gestión de las alertas se plantea para ser gestionada ya sea desde el escritorio, en el nodo principal, o a través de dispositivos móviles. La

gestión de las alertas en los dispositivos móviles permite tomar las medidas necesarias independientemente de restricciones de localización y tiempo.

En resumen, la característica distribuida de la arquitectura AIDeMaS provee la capacidad de optimizar el tiempo y los recursos disponibles durante las tareas críticas del proceso de detección de intrusiones.

A continuación se explica en detalle la arquitectura interna de los diferentes tipos de agentes de AIDeMaS. Para hacer una descripción detallada de la arquitectura propuesta se presentan tres sub-secciones: en la sub-sección 4.1.2 se describe la arquitectura interna de los tipos de agentes, en la sub-sección 4.1.3 se describe la arquitectura funcional de AIDeMaS, y en la sub-sección 4.1.4 se explica la arquitectura del sistema multi-agente tomando como referencia la arquitectura RETSINA (Sycara*, et al.*, 2003).

4.1.2 Arquitectura de Agentes

En esta sub-sección se analiza la estructura y las principales características de los tipos de agentes que componen la arquitectura propuesta en el apartado 4.1. Los agentes CBR-BDI (Bajo*, et al.*, 2010), (Pinzón*, et al.*, 2009a), (Carrascosa*, et al.*, 2008), (Corchado*, et al.*, 2003) constituyen el componente clave de la arquitectura, aportando las principales características diferenciales de AIDeMaS con respecto a las alternativas existentes.

Los agentes desarrollados en el marco de esta investigación deben proporcionar los mecanismos adecuados para llevar a cabo la tarea de detección de intrusiones de una manera eficiente y eficaz. En este sentido, se debe disponer de agentes capaces de monitorizar y capturar las peticiones de los usuarios, agentes capaces de auditar las actividades de usuario y detectar amenazas, agentes con capacidad para controlar la distribución de las tareas y la comunicación en las distintas capas, agentes con capacidad para llevar a cabo procesos de verificación y análisis sintáctico, agentes capaces de razonar, aprender y detectar anomalías, agentes capaces de administrar los recursos y gestionar las operaciones del directorio de agentes de la arquitectura, y finalmente agentes con capacidad para gestionar alertas y proveer una interfaz de interacción entre el experto humano y la arquitectura. De esta manera se han definido ocho tipos de agentes dentro de la arquitectura AIDeMaS. A continuación se presentan los ocho tipos de agentes identificados, describiendo la estructura interna de cada uno de ellos.

4.1.2.1 Agentes con Capacidades de Monitorización

Una de las principales fuentes de amenazas a la seguridad de la información tiene su origen en las peticiones de los usuarios. De la misma forma que se reciben peticiones de usuarios válidas, también se pueden recibir peticiones maliciosas de intrusos que se camuflan con el tráfico de las aplicaciones. Para detectar los intentos de intrusiones, el primer paso es llevar a cabo un proceso de monitorización y captura del tráfico generado por las peticiones de los usuarios. El proceso de monitorización es una de las tareas operativas que se realizan en el nivel más inferior de la arquitectura.

Para llevar a cabo la tarea de monitorización y captura del tráfico, se desarrolla un tipo de agente que captura el tráfico en la red y ejecuta un pre-procesamiento para extraer la carga útil de las peticiones HTTP e información de las cabeceras de las peticiones. El agente para realizar la captura del tráfico está equipado con las herramientas JPCAP (Fujii, 2000) y una versión adaptada de la herramienta TCPMon (Singh, 2010). Parte de la información extraída es almacenada en una base de datos para aplicar posteriormente estrategias de auditoría en busca de actividades maliciosas. Finalmente, la carga útil extraída y la información de las cabeceras de las peticiones se envían a la Capa Clasificación Ligera para su evaluación.

Este agente es un tipo de agente reactivo, donde su funcionalidad se limita a la captura de tráfico y extracción de información útil. Dependiendo de la carga de trabajo en la capa puede existir más de una instancia de este tipo de agente.

4.1.2.2 Agentes con Capacidad de Auditaría

Muchos de los enfoques de detección basan su estrategia en una inspección individual de las conexiones de los usuarios, centrando únicamente el análisis sobre el conjunto de características consideradas relevantes, sin tener la capacidad para encontrar y evaluar posibles relaciones dentro de las actividades de los usuarios. Algunos intrusos aprovechan esta limitación y construyen ataques a partir de un conjunto de acciones maliciosas, que sólo pueden ser detectados mediante la correlación de eventos. Partiendo de este conocimiento, se ha considerado necesario incorporar a AIDeMaS un tipo de agente para llevar a cabo un proceso de auditoría aplicando métodos estadísticos sencillos que no demanden un consumo significativo de recursos y permita detectar este tipo de comportamientos.

Este tipo de agente lleva a cabo un proceso de auditoría a partir de la información principalmente extraída de las conexiones de los usuarios. El formato de datos utilizado en el análisis es el *log* de las conexiones, siendo el formato más popular y que facilita obtener campos más descriptivos (Brugger, 2004). Mediante este proceso de auditoría se intenta identificar actividades maliciosas similares a envíos masivos de peticiones o ataques de tipo escáner de puertos. Muchos de los campos (esenciales, secundarios y calculados) considerados para ejecutar el proceso de auditoría han sido seleccionados a partir de estudios previos en la detección de intrusiones mediante la aplicación de técnicas del campo de la minería de datos (Brugger, 2004), (Dokas*, et al.*, 2002). Sin embargo, para el proceso de auditoría que nos interesa, sólo se han considerado algunos de los campos (*source_IP*, *destination_IP*, *source_port*, *destination_port*, *timestamp*, *time_connection*) presentados en la Tabla 4.1, dejando el resto de los campos como parámetros de entradas para procesos de auditoría más completos, propuestos como trabajo futuro. Adicionalmente, también se lleva un registro del estatus (válido, malicioso) de los usuarios, en concreto, de las direcciones IP utilizadas para el envío de peticiones. Con esta estrategia se logra prever ataques de usuarios clasificados como maliciosos o sospechosos.

Tabla 4.1. Campos considerados para el proceso de auditoria

Descripción	Campo
Tipo de Protocolo	str_protocol
Dirección del remitente	source_IP
Dirección del destinatario	destination_IP
Puerto del remitente	source_port
Puerto del destinatario	destination_port
Marca de tiempo	timestamp
Duración de la conexión	time_connection
Porcentaje de paquetes de datos	per_data_packets
Porcentaje de paquetes de control	per_control_packets
Número de bytes transferidos al destino	number_bytes_source
Número de paquetes	number_packets
Número de paquetes con la banderas FIN	number_pFIN
Número de paquetes con la banderas RST	number_pRST
Número de paquetes con la banderas SYN	number_pSYN
Número de conexiones realizadas por la misma fuente similar al registro actual en los últimos *n* segundos.	count_source
Número de peticiones al mismo servicio por la misma fuente en los últimos *n* segundos	count_service_source
Número de errores de establecimiento de conexión	number_error_con

Número total de conexiones	number_connection_src

Este agente con capacidad de auditoría aporta nueva funcionalidad, añadiendo un bloque de detección en el nivel más inferior de la arquitectura. Como resultado se adquiere la capacidad de detectar ataques complejos, estructurados a partir de un conjunto de acciones maliciosas, y además se dispone de la capacidad de anticipar intrusiones reconociendo la interacción con usuarios maliciosos.

Este tipo de agente ha sido diseñado utilizando una arquitectura BDI, pero sólo se han desarrollado sus funciones básicas dejando abierta la posibilidad de extender en un futuro la capacidad del agente, por ejemplo, incorporando mecanismos más complejos de auditoría utilizando técnicas de minería de datos como clasificación, agrupamiento, análisis de secuencia, etc. (Helali, 2010). Esto se propone como un trabajo futuro de investigación.

4.1.2.3 Agentes con Capacidad de Control

La arquitectura AIDeMaS se ha diseñado con una estructura jerárquica, con distintos tipos de agentes en cada una de las capas que la componen. Además, en algunas de las capas pueden requerirse más de una instancia de un tipo de agente dependiendo de la carga de trabajo. Esto requiere un mecanismo para controlar la comunicación de los agentes en cada capa y controlar la distribución de las tareas.

Para llevar a cabo las tareas de control, la arquitectura AIDeMaS dispone de un tipo de agente Control. Este agente es el encargado de gestionar la comunicación interna de la capa donde se sitúa, y también gestiona la comunicación externa. Esto quiere decir que toda la comunicación que entra o sale de la capa pasa por el agente Control. Este agente gestiona una lista local de los agentes activos en la capa y conoce el estatus (activo, inactivo) en todo momento, de cada uno de ellos enviando un mensaje *ping* el cual es devuelto con el estatus actual. Conociendo el estatus de los agentes se puede determinar la necesidad de solicitar nuevas instancias de los agentes cuando la carga de trabajo sea elevada. Adicionalmente, el agente control supervisa el comportamiento de los agentes de la capa permitiendo identificar agentes que hayan podido ser comprometidos durante la ejecución de sus tareas. En la Tabla 4.2 se presenta los parámetros mediante el cual se lleva un registro del comportamiento de los agentes activos. Inicialmente todos los agentes tienen asignada una valoración con un nivel de confianza alto el cual se modifica dependiendo del comportamiento del agente. Este nivel de confianza se modifica en función del rendimiento del agente durante la ejecución de las tareas asignadas

(*time_lasttask, average_time_agent*, *time_task_test).* Para todos los agentes activos es conocido un tiempo máximo para la ejecución de las tareas asignadas, lo que supone la capacidad para evaluar la confianza de los agentes. En los casos donde se identifique un agente comprometido, por ejemplo, un agente que haya sobrepasado el tiempo máximo para la ejecución de una tarea, inmediatamente se solicita a la capa Administración la eliminación de la instancia del agente malicioso.

Tabla 4.2. Campos para la supervisión de los agentes

Descripción	Campo
Estatus del agente (activo, inactivo)	*int_status*
Nivel de confianza (0= bajo, 1= alto)	*int_trust*
Tiempo sin actividad	*time_idle*
Tiempo utilizado en la ejecución de la última tarea	*time_lasttask*
Tiempo máximo del agente	*average_time_agent*
Tiempo utilizado en la ejecución de la tarea en el momento de la comprobación de estado.	*time_task_test*

Finalmente, el agente Control (de la capa Clasificación Ligera) es quien solicita la ejecución de la segunda fase del mecanismo de clasificación en aquellos casos donde no se obtuvo una solución definitiva. Ambas capas del mecanismo de clasificación envían sus resultados de la clasificación al agente Control correspondiente, para que este último envíe a su vez la solución a la capa Administración.

Este tipo de agente ha sido diseñado utilizando una arquitectura BDI, pero sólo se han desarrollado sus funciones básicas dejando abierta la posibilidad de extender la capacidad del agente, por ejemplo, incorporando mecanismos de planificación de tareas que permitan una asignación de tareas de manera inteligente.

4.1.2.4 Agentes con capacidad de Análisis

La carga útil de las peticiones de los usuarios enviadas a las aplicaciones y los servicios vienen codificadas, por ejemplo, en el estándar XML como es el caso de las solicitudes enviadas a los servicios Web, o una cadena SQL en el caso de las consultas a las base de datos. La detección de intrusiones en este tipo de peticiones requiere examinar además de la información de las cabeceras de las peticiones, el propio contenido de la petición. Para examinar el contenido y la

estructura sintáctica generalmente se utiliza algún tipo de analizador sintáctico/semántico. Un analizador sintáctico convierte el texto de entrada en otras estructuras (comúnmente árboles), que son útiles para el posterior análisis.

Para llevar a cabo esta tarea se ha diseñado un tipo de agente analizador, diseñado para emplear las herramientas adecuadas dependiendo de la carga útil a procesar. Este tipo de agente incorpora el analizador SAX (*Simple API for XML*) (Brownell, 2002) para el análisis de contenido XML. SAX fue la primera API para XML ampliamente adoptada en Java y más tarde implementada en muchos otros entornos de programación. A diferencia de otros analizadores, SAX funciona por eventos y métodos asociados, haciéndolo más eficiente desde el punto de vista computacional. A medida que el analizador va leyendo el documento XML y encuentra (los eventos) los componentes del documento (elementos, atributos, valores, etc.) o detecta errores, va invocando a las funciones que ha sido configuradas. En el caso de las peticiones a las bases de datos basadas en cadenas SQL, se implementa un analizador construido utilizando las herramientas *JFlex* y *Cup*. Ambas herramientas están desarrolladas en Java y se caracterizan por facilitar el análisis léxico y la escritura de la gramática. Debido a los distintos estándares SQL disponibles, se ha adoptado el gestor de base de datos MySQL 5.0–SQL Sin embargo, se puede extender fácilmente la funcionalidad del analizador a otros gestores de bases de datos.

Para este tipo de agente se ha utilizado la arquitectura BDI, considerando la posibilidad de extender sus funcionalidades, como es el caso de la detección de errores. Sin embargo, de momento sólo se ha desarrollado sus funciones básicas.

4.1.2.5 Agentes con Capacidades Avanzadas para Detectar Anomalías

Una parte fundamental de este trabajo es la utilización de agentes con capacidades avanzadas de razonamiento y aprendizaje para la detección de intrusiones en entornos dinámicos y distribuidos. Las técnicas de ataque evolucionan continuamente, lo que causa un serio problema para los mecanismos de detección de intrusiones existentes. Mantener la efectividad en la detección a lo largo del tiempo, resulta difícil si no se dispone de un mecanismo capaz de aprender y adaptarse a los cambios de comportamiento de las intrusiones de forma dinámica. Es por ello que la propuesta presentada en este trabajo de tesis supone un gran avance con respecto al estado del arte.

En este sentido se han diseñado tipos especiales de agentes dentro de la arquitectura AIDeMaS, y en concreto para llevar a cabo la identificación de anomalías se han incorporados dos tipos de agentes con capacidades novedosas. La combinación de ambos agentes permite realizar una clasificación en dos

etapas, necesaria dadas las restricciones de tiempo características en este tipo de entornos. El primer tipo de agente lleva a cabo la tarea de detección implementando un componente de clasificación ligera que permite realizar una clasificación rápida. El segundo tipo de agente utiliza un componente de clasificación pesado que permite realizar una clasificación más minuciosa, pero tiene un alto coste computacional y requiere de un mayor tiempo de respuesta.

Los tipos de agentes diseñados son agentes deliberativos BDI que hacen uso de sistemas CBR, también conocidos como agentes CBR-BDI (Bajo*, et al.*, 2010), (Pinzón*, et al.*, 2009b), (Pinzón*, et al.*, 2009c), (Pinzón*, et al.*, 2009a), (Pinzón*, et al.*, 2008a), (Pinzón*, et al.*, 2008b), (Carrascosa*, et al.*, 2008), (Corchado*, et al.*, 2008b), (Bajo*, et al.*, 2007), (Corchado*, et al.*, 2003). A continuación se presenta una formulación que permite relacionar los conceptos que definen la estructura interna y el comportamiento de un agente BDI con los conceptos manejados en sistemas de razonamiento basado en casos.

Se denomina Θ al conjunto que describe el entorno del agente BDI. Se llama $T(\Theta)$ al conjunto de atributos $\{\tau_1, \tau_2, ..., \tau_n\}$ en que el mundo es expresado.

- Definición: Una creencia b sobre Θ es una m-tupla de atributos de $T(\Theta)$. $b = \{\tau_1, \tau_2, ..., \tau_m\} \quad donde \quad m \le n$.
- Definición: Se llama conjunto de creencias sobre Θ y se denota por $\zeta(\Theta)$ a $\zeta(\Theta) = \left\{ b = \{\tau_1, \tau_2, ..., \tau_j\} \quad donde \quad j = 1, 2, ..., m \le n \right\}$.
- Definición: Se introduce el operador *"Λ de accesibilidad"* entre m creencias $b = \{b_1, b_2, ..., b_m\}$, y se denota $\Lambda[b_1, b_2, ..., b_m] = b_1 \wedge b_2 \wedge ..... \wedge b_m$, al operador que crea nuevas estructuras uniendo creencias compatibles entre sí. $\wedge$ es el operador lógico *and* sobre creencias, que no son otra cosa que expresiones lógicas.
- Definición: Se dice que el operador actuando sobre creencias es nulo si alguna de ellas no es accesible y se denota como $\Lambda[b_1, b_2, ..., b_m] = 0$.
- Definición: Una intención i sobre Θ es una s-tupla de creencias compatibles entre sí, $\left\{ i = (b_1, b_2, ..., b_m), \quad donde \quad \forall i, j \quad \Lambda[b_i, b_j] \neq 0, \quad s \in N \right\}$.
- Definición: Se llama conjunto de intenciones sobre Θ y se denota $I(\Theta)$ a $I(\Theta) = \left\{ (b_1, b_2, ..., b_k), \quad donde \quad k \in N \right\}$.

Una vez planteadas estas definiciones, se supone que a partir del análisis de un problema, se determinan un conjunto de parámetros que constituyen una base de atributos sobre la que estructurarlo.

En cuanto a un sistema CBR (Aamodt y Plaza, 1994), (Corchado y Laza, 2003) se puede definir una formulación para los componentes del sistema como sigue:

- Definición: Un conjunto de bases de casos (β). Una base de casos $B \in \beta$ es un conjunto finito de casos que se encuentra indexado. Una base de casos se define como una tupla ($\{c_1,c_2,\ldots, c_n\}$, ι). $\{c_1,c_2,\ldots, c_n\}$ son los casos que componen la base de casos y ι es el conjunto finito de características que permite indexar los casos.
- Definición: Un caso (c) representa una experiencia pasada. Un caso se representa mediante una secuencia de estados del entorno: *c={estado_incial, {acción x[estado_intermedio]}⁺, estado_final}*. Cada estado se representa a través de un conjunto de atributos que define el entorno en el que se encuentra situado el sistema CBR. Los estados se dividen en tres grupos:
    - Conjunto de estados iniciales (*estado_ini*), que representan la descripción del problema que debe ser resuelto
    - Conjunto de estados intermedios (*estado_inter*), que describen los distintos estados por los que pasa el entorno antes de alcanzar el estado final
    - Conjunto de estados finales (*estado_final*), que representa la descripción del entorno una vez que se ha alcanzado los objetivos iniciales.

Además de estados un caso contiene acciones, que representan el conjunto de acciones aplicadas a cada uno de los estados. Se definen mediante un nombre y un conjunto de argumentos.

- Definición: Un conjunto finito de atributos (k) son un conjunto de propiedades que permiten describir un estado.

- Definición: Un conjunto de índices (I) es un conjunto de características de ι, con ι incluido en k.

- Definición: Un conjunto de funciones de similitud (A) permiten determinar el grado de similitud que existe entre un problema a resolver y un caso.

El ciclo del sistema CBR queda definido a través de la Tabla 4.3. En la Tabla 4.3 es posible apreciar como en la etapa de recuperación se obtiene de la memoria de casos B aquellos casos $c_1,c_2,\ldots, c_k$, con una descripción de problema más similar al problema actual st_n utilizando para ello una métrica A. Se divide en dos etapas, la de indexación y la de selección. En la etapa de reutilización se obtiene una primera solución ($\{act_{ni}, \{st_{inter\_ni}\}\}^+, st_{final\_n}$) a partir de los casos recuperados y de la descripción del problema. En la etapa de revisión se evalúa la validez de la solución propuesta para, finalmente, en la etapa de aprendizaje aprender de la nueva experiencia.

Tabla 4.3. Ciclo de un sistema CBR

Recuperación	$(c_1,c_2,\ldots, c_k, A) \leftarrow Retrieve\ (st_n,B)$

	with $c_k = \{st_k, \{act\_{ki}, \{st_{inter\_ki}\}^*\}^+, st_{fin\_k}\}$, $k>0$ *and* $i>0$
Reutilización	$(st_n, \{act\_{ni}, \{st_{inter\_ni}\}^*\}^+, st_{final\_n}) \leftarrow$ *Reuse* $(st_n, (c_1, c_2, \ldots, c_k), A)$
Revisión	$(st_n, \{act\_{ni}, \{st_{inter\_ni}\}^*\}^+, st_{final\_n}) \leftarrow$ *Revision* (st_{fin_n})
Aprendizaje	$(st_n, \{act\_{ni}, \{st_{inter\_ni}\}^*\}^+, st_{final\_n}, B) \leftarrow$ *Learning* (c_n, B)

De esta forma, cuando un agente necesite resolver un problema, utilizará sus creencias, deseos e intenciones para construir una solución. Los deseos previos, junto con creencias e intenciones se almacenan en forma de casos como se muestra a continuación:

Caso: <Problema, Solución, Resultado>	**Agente BDI**
Problema: estado_inicial	Creencia: estado
Solución: secuencia de <acción, [estado_intermedio]>	Intención: secuencia de <acción>
Resultado: estado_final	Deseo: conjunto de <estado_final>

Una creencia es un estado que puede ser inicial (representa el problema que debe ser resuelto), intermedio (representa un estado intermedio del problema por el que se pasa antes de alcanzar el estado final) o final (representa el resultado obtenido después de partir del estado inicial y realizar un conjunto de acciones). Cada creencia posee una serie de atributos que la describen. Un deseo es un conjunto de estados finales. Dependiendo del problema un deseo podrá estar formado por uno o más estados finales que el agente quiere alcanzar.

Una intención es una secuencia ordenada de acciones. Las acciones son operaciones que se realizan sobre un determinado estado. Cada acción se define a través de un nombre y de un conjunto de argumentos. Así pues, en función de la información mostrada, se puede ver que existe una relación entre la descripción del caso del CBR y las creencias del modelo BDI.

Los agentes CBR-BDI, como se ha indicado anteriormente, son un componente clave de la arquitectura AIDeMaS. Son agentes con capacidades mejoradas que permiten aprender y razonar para adaptarse a los cambios de comportamientos de los ataques. Sin embargo, adicional a estas capacidades, los agentes CBR-BDI construidos para trabajar en la arquitectura AIDeMaS han sido diseñados para incorporar técnicas del aprendizaje automático que les dota de capacidades para generalizar comportamientos. La técnica utilizada se incorpora en la estructura interna del agente CBR-BDI, específicamente en la fase de reutilización del ciclo CBR, ejecutando una clasificación del caso bajo estudio que luego se convertirá en la solución final.

Estos dos tipos de agentes CBR-BDI incorporados en el mecanismo de clasificación de la arquitectura AIDeMaS son explicados de manera detallada más adelante.

4.1.2.6 Agentes con Capacidad de Gestión

Cada uno de los agentes en la arquitectura AIDeMaS tiene asignados roles y un conjunto de tareas a ejecutar. Dependiendo de la carga de trabajo en un determinado momento, se requiere gestionar el número de agentes necesarios para ejecutar el proceso de evaluación de todas las peticiones de usuarios recibidas. Para ello es necesario incorporar un componente para gestionar y controlar el funcionamiento global de la arquitectura.

Para llevar a cabo la gestión de la arquitectura, se ha incorporado un tipo de agente con capacidades de gestión. Este tipo de agente se sitúa en la capa Administración y lleva un registro de todos los agentes activos. Para ello mantiene una comunicación constante con los distintos agentes que controlan las capas de la arquitectura. Dependiendo de la carga de trabajo, este agente coordina, junto con los agentes de control de las distintas capas, la necesidad de crear nuevas instancias de agentes o la eliminación de agentes ociosos.

Para este tipo de agente, igualmente se ha utilizado la arquitectura BDI, previendo la posibilidad de extender sus funcionalidades para dotar con un mecanismo más eficiente el proceso de mantenimiento de la arquitectura.

4.1.2.7 Agentes con Capacidad de Interfaz

Un componente importante en cualquier tipo de arquitectura o aplicación es la capacidad para interaccionar con usuarios del sistema. Además, generalmente en este tipo de aplicaciones la toma de decisiones de alto nivel tiene como último responsable a un experto para garantizar la fiabilidad de las acciones o respuestas y la estabilidad del sistema.

En la arquitectura AIDeMaS se ha considerado un tipo de agente especializado en proporcionar capacidades de interacción con los usuarios del sistema. Este tipo de agente se sitúa en la capa Administración y proporciona una interfaz entre la arquitectura y el encargado o experto humano. Este tipo de agente se ha diseñado para trabajar en distintitos tipos de dispositivos móviles tales como PDAs, teléfonos móviles u ordenadores portátiles. Esta capacidad dota a la arquitectura de ubiquidad en el acceso al sistema, permitiendo que la persona encargada pueda recibir las alertas de intrusiones o facilitar su intervención independientemente de su localización.

Mediante este tipo de agente se facilita una interfaz para evaluar las peticiones sospechosas que no han podido ser clasificadas de forma directa en las capas inferiores. El agente dispone de herramientas y se apoya con los agentes de las capas inferiores para proporcionar al experto humano toda la información necesaria para resolver las peticiones. En la Tabla 4.4 se presenta la información obtenida de las capas inferiores de la arquitectura que se muestra al experto.

Tabla 4.4. Información obtenida de las capas para apoyar en la toma de decisiones

Capa/Agente	Información
Monitorización / agAuditor	Número de peticiones en n tiempo
Monitorización / agAuditor	Estatus del usuario (válido/malicioso)
Clasificación Ligera/agControl2	Resultado de la clasificación
Clasificación Pesada/agAnalizador	Tiempo de análisis sintáctico
Clasificación Pesada /agAnalizador	Número de errores de validación
Clasificación Pesada /agAnalizador	Resultado de la clasificación

Dependiendo de la configuración de la arquitectura, se tiene la posibilidad que el experto ejecute las acciones necesarias cuando se reciba una alerta de intrusión desde las capas inferiores, o en caso contrario, se aplique las medidas necesarias que indica la configuración por defecto de la arquitectura. Finalmente, este tipo de agente puede comunicarse con todos los agentes en las diferentes capas de la arquitectura para ajustes en la configuración.

Similar al resto de los agentes de la arquitectura, este tipo de agente se basa en el modelo CBR-BDI, para dejar abierta la posibilidad de extender sus funcionalidades de cara a incorporar nuevos mecanismos más eficientes e inteligentes, por ejemplo mecanismos de visualización (Herrero*, et al.*, 2009) para facilitar al experto la identificación y confirmación de actividades anómalas.

4.1.3 Arquitectura Funcional

Una arquitectura funcional describe los diferentes bloques funcionales que componen un sistema y las relaciones que se establecen entre ellos. Los sistemas multi-agente se estructuran teniendo en cuenta modularización dentro de un sistema, pero también la reutilización, integración y rendimiento. Algunos de los problemas de integración aparecen por la incompatibilidad entre plataformas de agentes.

En el caso de la arquitectura AIDeMaS, la arquitectura funcional se plantea a través de las capas. Cada capa de la arquitectura podría verse como un bloque funcional donde se agrupan agentes dependiendo de las tareas a ejecutar. Los agentes se comunican entre sí enviando la información de las capas inferiores hacia las capas superiores para determinar la validez de las peticiones de usuarios. En el caso de la gestión de la arquitectura, la comunicación se lleva a cabo desde la capa Administración, la capa superior de la arquitectura, a las capas inferiores. Finalmente a nivel de control, cada capa maneja una comunicación interna para la realización de las tareas. En la Figura 4.3 se puede apreciar la arquitectura funcional y los distintos bloques (capas de la arquitectura).

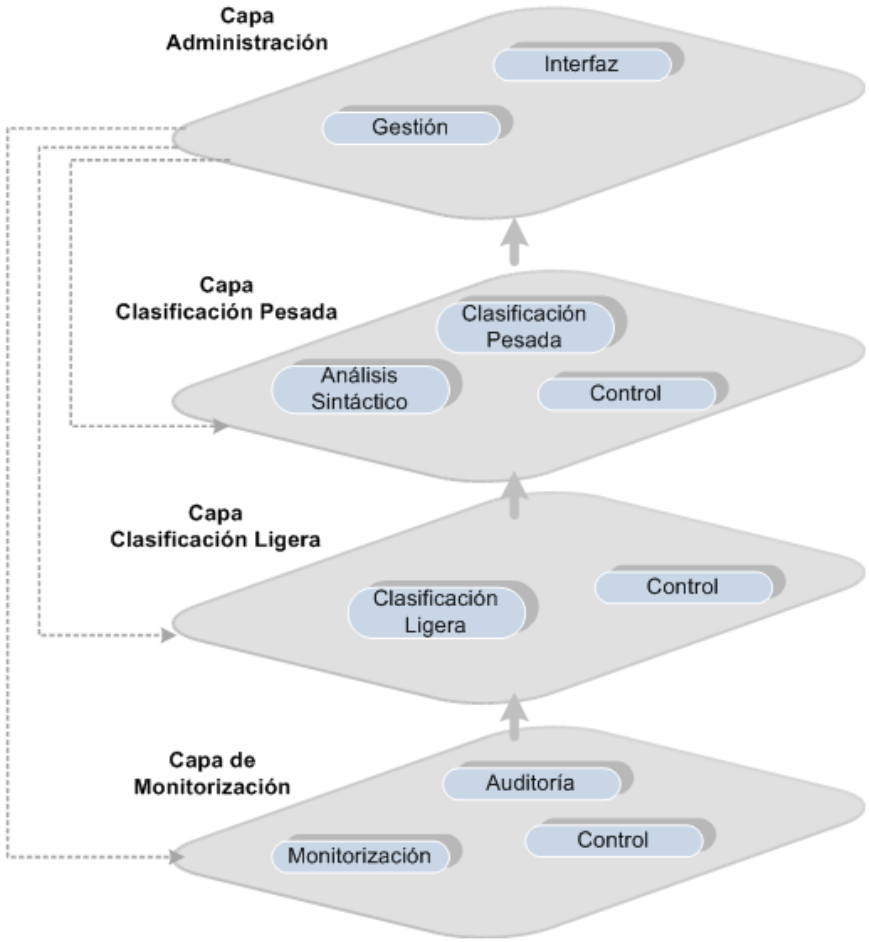

Figura 4.3. Arquitectura funcional y los distintos bloques funcionales (Capas)

En cada una de las capas de la arquitectura intervienen distintos tipos de agentes pero son los agentes de control los que se encargan de coordinar tanto la comunicación interna en las capas así como la comunicación externa. En el caso de un evento para la gestión del directorio de agentes, la comunicación se origina en la capa superior de la arquitectura y desciende hasta la capa inferior requerida, solicitando los servicios del agente control asignado a la capa. Finalmente, otro evento de comunicación se origina producto de la modificación en la configuración de los agentes. En este caso la comunicación se inicia en la capa superior y se establece directamente con el agente requerido, sin la intervención del agente control.

4.1.4 Arquitectura del Sistema Multi-Agente

La arquitectura del sistema multi-agente queda definida a través de la arquitectura de agentes, de la arquitectura funcional y de la estructura tanto del sistema multi-agente como de cada agente individual.

En la sección 4.1.2 se explicó la estructura interna de los tipos de agente de la arquitectura pero no se presentaron los agentes en concreto. A continuación se describen cada uno de los agentes incorporados en las distintas capas de la arquitectura.

- Agentes agMonitores: Monitorizan el tráfico en la red, extraen la carga útil (consultas SQL, mensajes SOAP) de las peticiones HTTP e información de las cabeceras de las peticiones. Para realizar estas tareas, se utiliza la librería JPCAP (Fujii, 2000) y se adaptaron clases de la herramienta TCPMon (Singh, 2010), una utilidad de código abierto para monitorizar conexiones TCP. Parte de la información extraída (dirección remitente, puerto del remitente, dirección destino, puerto destino, tipo de solicitud, fecha y hora de la solicitud) es almacenada en una base de datos para una evaluación posterior. Finalmente, la carga útil extraída y la información de las cabeceras de las peticiones es enviada al agente control de la Capa Monitorización para reenviarla a la siguiente capa de la arquitectura.

- Agente agAuditor: Lleva a cabo un proceso de auditoría a partir de información extraída de las conexiones de los usuarios. Mediante este proceso de auditoría se intenta identificar actividades maliciosas similares a envíos masivos de peticiones o ataques de tipo escaneo. Los campos considerados incluyen: source_IP, destination_IP, source_port, destination_port, timestamp, time_connection presentados en la Tabla 4.1. También se lleva un registro del estatus (válido, malicioso) de los usuarios,

en concreto, de las direcciones IP utilizadas para el envío de peticiones. Con esta estrategia se logra bloquear a usuarios con un perfil malicioso.

- Agente agControlC1: Lleva a cabo las funciones de supervisión y coordinación de la comunicación y la distribución de las tareas de la capa 1. Se encarga de redirigir la información enviada por el agente agMonitor a la capa Clasificación Ligera. Supervisa el funcionamiento del agente agMonitor y agAuditoria, y determina la necesidad de crear nuevas instancias o eliminar agentes comprometidos u ociosos.

- Agentes agClasLigero: Se encarga de ejecutar la primera etapa del mecanismo de clasificación detectando intrusiones conocidas. Es uno de los tipos de agentes basados en el modelo CBR-BDI (Pinzón*, et al.*, 2009b), (Pinzón*, et al.*, 2009c), (Pinzón*, et al.*, 2009a), (Pinzón*, et al.*, 2008a), (Pinzón*, et al.*, 2008b), (Carrascosa*, et al.*, 2008), (Corchado*, et al.*, 2008b), (Bajo*, et al.*, 2007), (Corchado*, et al.*, 2003). Incorpora en su estructura interna un motor de razonamiento basado en casos y a su vez una técnica de aprendizaje automático poco costosa computacionalmente con resultados fiables. Una vez resuelta la clasificación se envían los resultados al agente agControlC2. Dependiendo de la carga de trabajo, puede existir más de una instancia del agente agClasLigero.

- Agente agControlC2: Controla y coordina el trabajo de clasificación en la capa 2 de la arquitectura. Recibe la información de la capa inferior y asigna la tarea de clasificación a un agente agClasLigero. Dependiendo del resultado de la clasificación determina si se requiere ejecutar la segunda etapa del mecanismo de clasificación, y envía los resultados a la capa Administración. En el caso que sea necesario ejecutar la segunda etapa del mecanismo de clasificación el agente agControlC2 envía todos los datos necesarios provenientes de la capa Monitorización. Finalmente, este agente mantiene comunicación con la capa superior y determina la necesidad de crear o eliminar instancias del agente agClasLigero.

- Agente agControlC3: Se encarga de coordinar las tareas de la capa 3. Entre sus tareas están recibir los datos para la clasificación y asignar un agente agAnalizador y un agClaspesado para ejecutar el proceso de clasificación. También se encarga de supervisar el funcionamiento de estos dos agentes principalmente del agente agAnalizador ya que este agente puede ser susceptible a amenazas debido a que las tareas de procesamiento y análisis pueden inducir algún tipo de ataque, por ejemplo un desbordamiento de memoria. En este caso se supervisa el tiempo requerido para la ejecución de la tarea y se compara con un umbral de tiempo establecido. Si el agente agControlC3 detecta un agente agAnalizador comprometido, se comunica con la capa Administración para eliminar la instancia del agente correspondiente. Finalmente, otra de las tareas del agente agControlC3 es recibir el resultado de la clasificación y enviarlo a la capa Administración.

- Agente agAnalizadores: Lleva a cabo el procesamiento y análisis de los datos enviados por el agente agControlC3. Extrae los datos de la petición utilizando herramientas de análisis léxico y sintáctico (*SAX, JFlex, Cup*). Los datos extraídos serán utilizados para ejecutar la segunda etapa de clasificación por el agente agClasPesado. Además, el agente agAnalizador se encarga de buscar e identificar patrones conocidos de ataque y aplicar políticas de validación de la petición. En caso de detectar patrones de ataque o si se producen errores, el agente informa del incidente al agente agControlC3 para que éste pueda activar las medidas necesarias dependiendo de la configuración de la arquitectura. Dependiendo de la carga de trabajo, puede existir más de una instancia del agente agAnalizador.

- Agente agClasPesado: Es el segundo tipo de agente CBR-BDI diseñado para ejecutar la segunda etapa del mecanismo de clasificación. Al igual que el agente agClasLigero, es uno de los tipos de agentes clave de la arquitectura. Incorpora un motor de razonamiento basado en casos y utiliza técnicas de aprendizaje automático para la clasificación de las peticiones. Pero a diferencia del agente agClasLigero, las técnicas de clasificación utilizadas suelen requerir un coste computacional significativo. Dependiendo de la configuración de la arquitectura, el agente agClasPesado puede clasificar todas las peticiones enviadas a la capa Clasificación, o sólo las peticiones previamente clasificadas como sospechosas por el agente agClasLigero. Dependiendo de la carga de trabajo, puede existir más de una instancia del agente agClasPesado.

- Agente agGestión: Maneja un directorio de todos los agentes activos (*Agent Management System*, AMS) siguiendo el estándar FIPA (FIPA, 2007). De igual forma se plantea un facilitador de directorio (*Directory Facilitator*, DF) como complemento para posibles aplicaciones futuras. Este agente se coordina con los distintos agentes que controlan las capas de la arquitectura. Dependiendo de la carga de trabajo, los agentes de control solicitan al agente agGestión la creación y registro de nuevas instancias de agentes o la eliminación de instancias de agentes ociosos.

- Agente agAdminGUI: Proporciona una interfaz entre la arquitectura y el encargado o experto humano. Puede trabajar en distintitos tipos de dispositivos móviles.

La Figura 4.4 presenta la arquitectura del sistema multi-agente con las distintas capas y agentes definidos.

La arquitectura AIDeMaS, en su parte de gestión, incorpora servicios de nombres de agentes (ANS) y de facilidad de directorio (DF), tal y como establece el estándar FIPA (FIPA, 2007) y de forma similar a como lo hacen otras arquitecturas como RETSINA o plataformas de agentes como JADE (Bellifemine, *et al.*, 2001).

La Figura 4.5 presenta la arquitectura desde el punto de vista de comunicación. La arquitectura es compatible con los estándares FIPA (*Foundation for Intelligent Physical Agents*). La especificación FIPA se centra fundamentalmente en las interacciones que deben realizarse entre los agentes. Así pues, se busca que exista una compatibilidad entre distintas plataformas de agentes en conceptos tales como arquitectura abstracta y comunicación. FIPA indica que las plataformas de agentes compatibles FIPA deben contener un sistema de gestión de los agentes tal que garantice que no hay dos agentes con el mismo nombre. Además exige un servicio de directorio en el que los agentes puedan registrar y buscar servicios. Se exige también un servicio de transporte de mensajes. Finalmente, FIPA exige una estructura de mensaje ACL ("*Agent Communication Language*").

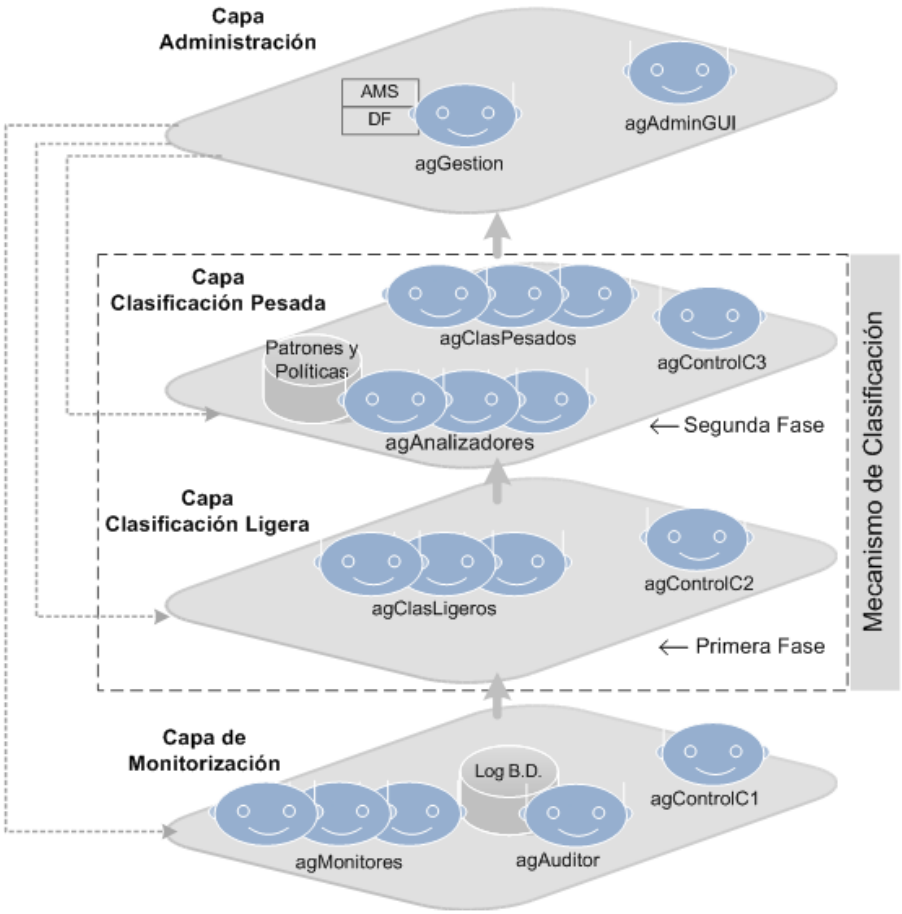

Figura 4.4. Arquitectura del sistema multi-agente

La arquitectura AIDeMaS contiene un agente agGestion que se ocupa de implementar un servicio MS ("*Management System*"), encargado de gestionar los agentes conectados a la plataforma (similar al agente AMS utilizado en JADE).

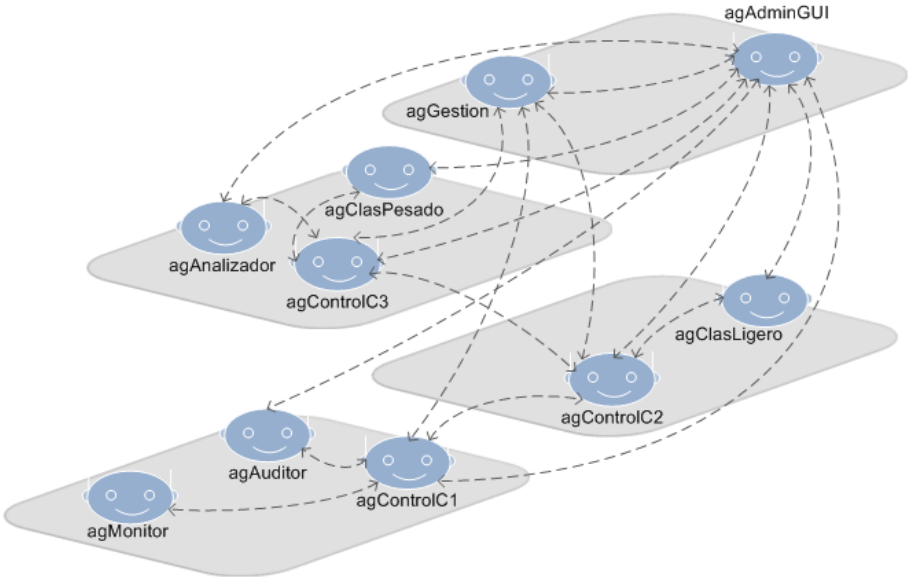

Figura 4.5. Arquitectura del sistema multi-agente desde el punto de la comunicación de los agentes

4.1.4.1 Comunicación de los Agentes

La comunicación de los agentes dentro de la arquitectura AIDeMaS se realiza siguiendo las especificaciones FIPA-ACL (FIPA, 2002), (Bauer y Huget, 2003) (Odell y Huget, 2003). Este estándar establece los estándares para el intercambio de mensajes a nivel de sistemas multi-agente permitiendo la interconexión entre diferentes sistemas. El estándar FIPA-ACL define una serie de parámetros, de los cuales algunos son requeridos en la estructura del mensaje y otros opcionales dependiendo de la configuración establecida. En la Tabla 4.5 se resumen los campos establecidos por el estándar FIPA-ACL.

Tabla 4.5. Parámetros establecidos por el estándar FIPA-ACL

Parámetro	Categoría
performative	Tipo de comunicación

sender	Emisor
receiver	Receptor
reply-to	Agente a responder
content	Contenido
language	Descripción del contenido
encoding	Descripción del contenido
ontology	Descripción del contenido
protocol	Control de la comunicación
conversation-id	Control de la comunicación
reply-with	Control de la comunicación
in-reply-to	Control de la comunicación
reply-by	Control de la comunicación

De los campos más importantes en la estructura del mensaje encontramos el campo *performative* que indica el tipo de comunicación, *language* indica la definición del lenguaje utilizado, *encoding* se refiere a la codificación o el formato del mensaje y por último tenemos el campo *ontology* que indica, la ontología u ontologías utilizadas para definir el significado de los símbolos incluidos en el contenido del mensaje. Una completa descripción de cada uno de los parámetros se puede encontrar en la especificación de FIPA-ACL (FIPA, 2002).

Para llevar a cabo la comunicación dentro de la arquitectura AIDeMaS, los mensajes FIPA-ACL, al ser objetos, requieren un sistema de comunicación basado en el paso de objetos, específicamente objetos de Java en nuestro caso. Esta comunicación se realiza utilizando JAVA RMI que permite la invocación de métodos en objetos remotos. A diferencia de la comunicación a través de HTTP que provee mayor compatibilidad, RMI suele ser más eficiente en cuanto a tiempos. Finalmente JAVA RMI permite una mayor compatibilidad en cuanto al desarrollo de la arquitectura al estar basada completamente en JAVA.

Finalmente, un aspecto importante en la arquitectura AIDeMaS es la seguridad. El principal problema de seguridad viene relacionado con la seguridad de la comunicación. Para resolver este problema, la seguridad en las comunicaciones se ve mejorada con la incorporación de mecanismos de autentificación y de seguridad SSL en RMI.

4.2 Mecanismo para la Detección de Intrusiones

La detección de intrusiones basada en la utilización de técnicas del aprendizaje automático está siendo ampliamente estudiada dentro del campo de los IDS debido a sus capacidades de generalización, que ayudan en la detección tanto de intrusiones conocidas como desconocidas (Mukkamala*, et al.*, 2005),

(Tsai, *et al.*, 2009). Algunas de las técnicas más aplicadas en la detección de intrusiones incluyen las redes neuronales artificiales, algoritmos genéticos, máquinas de soporte vectorial, lógica borrosa y técnicas de la minería de datos (Tsai, *et al.*, 2009), (Kandeeban y Rajesh, 2010). Las técnicas de aprendizaje automático suelen ser agrupadas en dos grandes categorías: supervisadas y no supervisadas. En el aprendizaje supervisado la estimación se realiza a partir de un conjunto de entrenamiento, del que se conocen tanto los datos de entrada como la clase a la que pertenecen. Generalmente el aprendizaje supervisado se asocia a la utilización de métodos de clasificación en la estrategia de detección de uso indebido para la identificación de ataques donde los patrones son conocidos. En cambio, en el aprendizaje no supervisado generalmente el sistema de clasificación de patrones debe diseñarse partiendo de un conjunto de patrones de entrenamiento para los cuales no se conocen sus etiquetas de clase. Esta estrategia por lo tanto resulta adecuada para llevar a cabo una detección de intrusiones basada en detección de anomalías permitiendo detectar ataques no conocidos. Sin embargo, en la medida que se pueda disponer de un conjunto de entrenamiento etiquetado, la detección por anomalía puede operar en tres modos distintos (Chandola, *et al.*, 2009):

- Detección de anomalía supervisada: Las técnicas entrenadas en modo supervisado asumen la disponibilidad de un conjunto de datos de entrenamiento, que tienen instancias etiquetadas para ambas clases: normal y anómala. Cualquier instancia de datos desconocida es validada contra un modelo para determinar a cual clase pertenece.

- Detección de anomalía Semi-Supervisada: Las técnicas que operan en un modo semi-supervisado asumen que el conjunto de entrenamiento tiene instancias etiquetadas sólo para la clase normal.

- Detección de anomalía No Supervisada: Las técnicas que operan en modo no supervisado no requieren de un conjunto de entrenamiento. Las técnicas en esta categoría hacen la suposición implícita que las instancias normales son mucho más frecuente que las anomalías en el conjunto de datos de prueba. Sin embargo, si esta suposición no es verdadera, entonces las técnicas sufren de tasas altas de falsas alarmas.

Por último, adicionalmente a las técnicas supervisadas y no supervisadas, hay que añadir las técnicas híbridas. El desarrollo de sistemas híbridos basados en la combinación de ambas estrategias se construyen con el objetivo de alcanzar una mayor exactitud en la detección de intrusiones (Tsai, *et al.*, 2009).

La estrategia de detección planteada en AIDeMaS es totalmente novedosa y se basa en una detección de anomalía supervisada donde se dispone de los conjuntos de entrenamientos, que tienen instancias etiquetadas para ambas clases. Esta estrategia de detección dota a la arquitectura la capacidad de evolucionar frente a nuevos ataques y a variantes en los ataques. No obstante,

AIDeMaS incorpora una estrategia de detección de uso indebido en la capa Clasificación Pesada como complemento al mecanismo de clasificación supervisada. El mecanismo de detección de uso indebido implementado se basa principalmente en la búsqueda y comparación de patrones conocidos de ataques durante el análisis sintáctico aplicado al contenido de las peticiones de usuarios. El análisis sintáctico se requiere en esta etapa del mecanismo de clasificación para la extracción de los parámetros de entrada del componente clasificador de anomalía, con lo cual, este proceso no representa una carga adicional significativa a la clasificación. Por el contrario, se añade otro componente para detectar ataques en la etapa inicial del mecanismo clasificador de la capa Clasificación Pesada. Este componente adicional ayuda a optimizar el tiempo de clasificación en aquellos casos donde se detecte un número de errores superior a un umbral establecido. Una vez que se contabiliza un número de errores establecidos, se detiene el proceso de clasificación y se envía una alerta a la capa Administración para tomar las medidas necesarias.

La principal estrategia de detección de AIDeMaS se basa en una detección por anomalía. El mecanismo de detección diseñado se lleva a cabo en las dos capas que integran el mecanismo de clasificación de la arquitectura. En la primera capa se lleva a cabo una clasificación ligera utilizando técnicas de aprendizaje automático poco costosas computacionalmente. En la segunda capa del mecanismo de clasificación se utilizan igualmente técnicas de aprendizaje automático pero que pueden demandar una cantidad significativa de tiempo y recursos. La Figura 4.6 presenta el mecanismo de clasificación en dos etapas.

Como se planteó al inicio de este capítulo, dos de los ataques abordados en este trabajo de tesis son los ataques de inyección SQL y los ataques XDoS. Esta delimitación de los ataques abordados permite conocer sus características, sus técnicas de ataques, su evolución o cambios de comportamiento, etc. A partir de estos datos se extrae información descriptiva del ataque que se utiliza para construir modelos que son capaces de aprender y distinguir las actividades maliciosas de las legítimas.

A continuación se explican los dos tipos de agentes CBR-BDI incorporados en el mecanismo de clasificación en dos etapas de la arquitectura AIDeMaS.

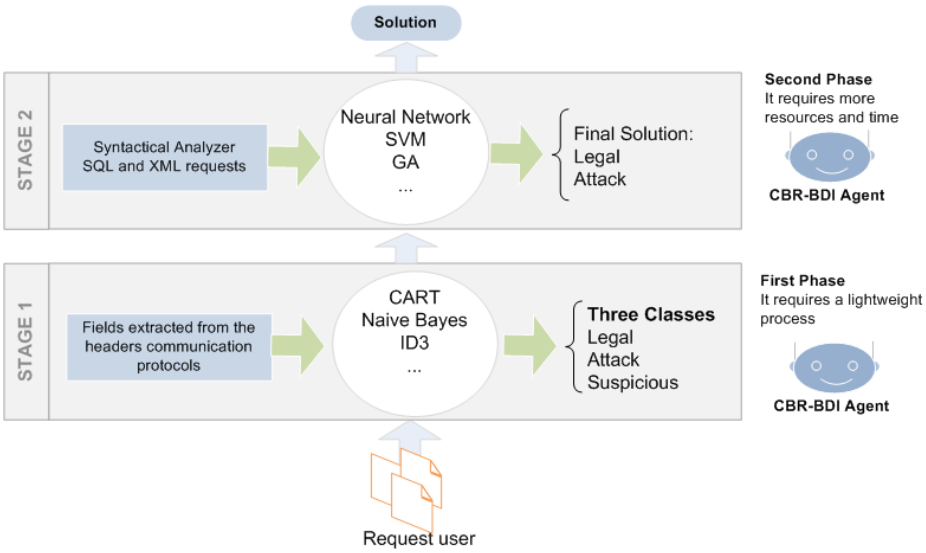

Figura 4.6. Mecanismo de clasificación en dos etapas

4.2.1 Clasificación Ligera

Este tipo de agente se incorpora en la primera etapa del mecanismo de clasificación para clasificar las peticiones de usuario mediante un modelo computacional ligero. La capacidad del agente CBR-BDI para determinar si una petición de un usuario es válida o maliciosa viene dada por la técnica de clasificación incorporada en la etapa de reutilización o adaptación del ciclo CBR.

Los datos utilizados por el clasificador son extraídos de los protocolos de comunicación utilizados para el intercambio de información entre el cliente y el proveedor. La extracción de los datos se realiza de forma rápida y no requiere una cantidad de recursos computacionales significativa. Los datos extraídos proporcionan información descriptiva para detectar ataques conocidos en los entornos de las aplicaciones. La Tabla 4.6 presenta a modo de ejemplo los campos descriptivos para el ataque XDoS, que ha sido considerado para una clasificación en dos etapas. Los ataques de inyección SQL se han planteado únicamente para ser resueltos en la segunda etapa del mecanismo de clasificación que se explicará más adelante.

Tabla 4.6. Campos descriptivos – Primera Etapa – Ataques *XDoS*

Descripción	Campo	Tipo
Identificación del servicio	IDService	*Int*
Máscara de subred	Subnet mask	*String*
Tamaño del mensaje	SizeMessage	*Int*
Número de saltos del mensaje	NTimeRouting	*Int*
Longitud del campo SOAPACTION	LengthSOAPAction	*Int*
Marco de tiempo transcurrido después del último mensaje.	TFMessageSent	*Int*

Con los campos disponibles se seleccionan las técnicas para la detección de actividades maliciosas en la capa Clasificación Ligera. En la Tabla 4.7 y Figura 4.7 se enumeran algunas de las técnicas evaluadas empíricamente para la capa Clasificación Ligera. Para analizar el tiempo de ejecución se utilizaron un total de 1500 peticiones de usuarios y se replicaron y analizaron 30,000 ejecuciones y se determinó el tiempo promedio de ejecución.

Tabla 4.7. Tiempo de las técnicas de clasificación

Classifiers	Mean Time	*Worst Case Estimated Time*
Decission Tree	9,86E-03	0,03790198
Naive Bayes	5,24E-03	0,01986013
SMO	1,33E-02	0,05122575
N. Network	0,287	1,07269979

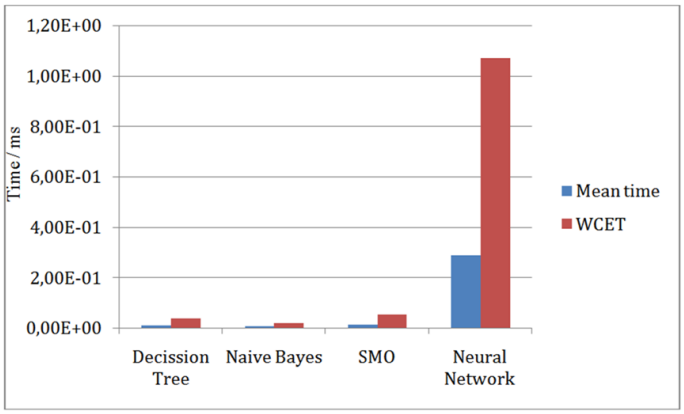

Figura 4.7. Tiempo promedio y peor tiempo de las técnicas de clasificación

Finalmente, las peticiones de usuarios, en este caso los mensajes SOAP, son capturados por los agentes especializados, extrayendo la carga útil y la información de las cabeceras del protocolo de comunicación. La información obtenida se envía al agente de control de la capa de Clasificación Ligera quien asignará un agente de clasificación para llevar a cabo la clasificación de la petición de usuario. El agente ejecuta la clasificación de las peticiones de usuario mediante la ejecución de un ciclo CBR. A continuación se explican brevemente las fases del ciclo CBR ejecutado para la clasificación de las peticiones de usuarios.

- Fase de Recuperación (*Retrieve*): En esta fase del ciclo CBR se dispone de una memoria de casos y una memoria de modelos donde se almacena un conjunto de casos que representan la experiencia previa del agente clasificador y los modelos de clasificación. Todos los casos similares al nuevo caso son recuperados de la memoria junto con los modelos que fueron usados para su clasificación. Una vez se han recuperados los casos similares, se ejecuta la fase de reutilización del ciclo CBR.

- Fase de Reutilización (*Reuse*): En esta fase se reutilizan los casos similares recuperados para generar una nueva solución para el caso presentado. Dependiendo de la técnica de clasificación utilizada y los modelos recuperados, los casos recuperados son utilizados en la etapa de aprendizaje para entrenar el modelo de clasificación actual. Con el modelo entrenado, se presenta el nuevo caso para su clasificación. Finalmente se ejecuta la siguiente fase para revisar la solución generada.

- Fase de revisión (*Revise*): En la fase de revisión, la solución generada para el nuevo caso puede ser revisada por un experto humano. En fase de prueba la revisión se lleva a cabo de modo *offline* permitiendo retroalimentar el modelo en aquellos casos clasificados de forma errónea.

- Fase de mantenimiento (*Retain*): Es la última fase del ciclo CBR donde se lleva a cabo el mantenimiento de la memoria de casos. Si el nuevo caso es diferente de los casos almacenados, es decir, el nuevo caso y su solución se almacenan en la base de casos para casos similares futuros. Durante esta etapa, también se actualizan los modelos de clasificación si se ha producido algún error de clasificación.

Finalizado el ciclo CBR se tiene la clasificación de la petición de usuario. Si la petición de usuario ha sido clasificada como ataque, finaliza el proceso de clasificación y se envía una alerta a la capa Administración. Si por el contrario la petición ha sido clasificada como sospechosa, entonces el agente Control de la capa Clasificación Ligera se comunica con el agente Control de la capa Clasificación Pesada para iniciar la ejecución de la segunda etapa del mecanismo de clasificación. Finalmente, si la petición de usuario ha sido clasificada como válida, se autoriza su ejecución en el servicio requerido.

4.2.2 Clasificación Pesada

El agente de clasificación agClasPesado es el segundo tipo de agente CBR-BDI que trabaja en la segunda etapa del mecanismo de clasificación para realizar una clasificación exhaustiva (pesada).

El agente agClasPesado trabaja en conjunto con otros agentes de la capa Clasificación Pesada para ejecutar la clasificación de la petición de usuario. El agente agControlC3 encargado de coordinar la comunicación y la asignación de tareas en la capa, y el agente agAnalizador encargado de ejecutar el proceso de análisis sintáctico dentro de la carga útil (contenido) de las peticiones de usuario.

Al igual que el agente AgClasLigero, el agente agClasPesado incorpora una técnica del campo del aprendizaje automático para llevar a cabo la clasificación de las peticiones de los usuarios. Esta etapa del mecanismo de clasificación se considera "pesada" debido a que se requiere ejecutar un análisis sintáctico exhaustivo de las peticiones de usuario, que generalmente demanda una cantidad de recursos computacionales significativa. Adicionalmente a este proceso, las técnicas de aprendizaje automático utilizadas para la clasificación pueden incurrir en una demanda de tiempo y recursos superior a lo requerido en la primera etapa del mecanismo de clasificación.

El procedimiento de clasificación de las peticiones de usuario en esta etapa se inicia cuando el agente agControlC3 recibe las peticiones de usuarios y asigna un agente agAnalizador y un agente agClasPesado para llevar a cabo la clasificación. El agente agAnalizador inicia el proceso de análisis del contenido extrayendo los datos de los campos requeridos como entradas del modelo de clasificación incorporado en el agente agClasPesado. Durante el proceso de análisis, el agente agAnalizador también ejecuta una búsqueda de patrones conocidos de ataque y errores sintácticos en el contenido. Si durante el proceso de análisis del contenido de la petición se detectan patrones conocidos de ataque o se supera el umbral del número de errores establecidos, se detiene el proceso de clasificación y se envía una alerta a la capa Administración para gestionar las medidas necesarias.

En las Tabla 4.8 y Tabla 4.9 se presentan los campos requeridos como parámetros de entradas del modelo de clasificación de la segunda etapa del mecanismo de clasificación para los ataques de inyección SQL y los ataques *XDoS.*

Tabla 4.8. Campos descriptivos – Segunda Etapa – Ataques de inyección SQL

Problem fields	Description
IP_Address	String

Affected_table	Integer
Affected_field	Integer
Command_type	Integer
Word_GroupBy	Boolean
Word_Having	Boolean
Word_OrderBy	Boolean
Numer_And	Integer
Numer_Or	Integer
Number_literals	Integer
Number_LOL	Integer
Length_SQL_String	Integer

Tabla 4.9. Campos descriptivos – Segunda Etapa – Ataques *XDoS*

Fields	Type
IDService	Int
MaskSubnet	String
SizeMessage	Int
NTimeRouting	Int
LengthSOAPAction	Int
MustUnderstandTrue	Boolean
NumberHeaderBlock	Int
NElementsBody	Int
NestingDepthElements	Int
NXMLTagRepeated	Int
NLeafNodesBody	Int
NAttributesDeclared	Int
CPUTimeParsing	Int
SizeKbMemoryParser	Int

Finalizado el análisis de la carga útil de la petición y obtenidos los parámetros de entrada del modelo construido se inicia el proceso de clasificación mediante la ejecución de un ciclo CBR que se detalla brevemente a continuación.

- Fase de Recuperación (*Retrieve*): Se recuperan de la memoria de casos todos los casos similares al nuevo caso en base al tipo de consulta y máscara de red. A su vez, se recupera de la memoria de reglas el clasificador anteriormente usado para la clasificación de los casos recuperados Con los casos y el clasificador recuperado se ejecuta la fase de reutilización del ciclo CBR.

- Fase de Reutilización (*Reuse*): En esta etapa se utiliza una técnica de clasificación, generalmente más costosa, para resolver la clasificación de las peticiones de usuario. En la etapa de aprendizaje se utilizan los casos similares recuperados para el entrenamiento del modelo (normalmente esta epata no se realiza ya que se reutiliza el clasificador previamente almacenado), y en la fase de prueba o funcionamiento se presenta el nuevo caso para su clasificación. Finalizada la clasificación, se determina la validez de la petición de usuario. Si la petición de usuario se clasificó como maliciosa o sospechosa, se envía una alerta a la capa Administración para tomar las medidas necesarias. En el caso que se haya clasificado como válida, se permite el paso de la petición de usuario para su ejecución en el servicio solicitado.

- Fase de revisión (*Revise*): La solución generada para el nuevo caso requiere una revisión generalmente llevada a cabo por un experto humano. Durante la fase de funcionamiento la revisión se lleva a cabo de modo *offline* permitiendo retroalimentar el modelo par los casos donde se realizó una clasificación errónea.

- Fase de mantenimiento (*Retain*): Si el nuevo caso es diferente a los almacenados en la base de casos, se almacena en la base de casos como nueva experiencia para la resolución de problemas futuros. En caso de clasificación errónea se actualiza offline el clasificador asociado a los casos recuperados.

Tanto la capa Clasificación Ligera como la capa Clasificación Pesada incorporan técnicas de clasificación del campo del aprendizaje automático, pero a diferencia de las técnicas utilizadas en la capa Clasificación Ligera, las técnicas de la capa Clasificación Pesada generalmente incurren en una demanda significativa de recursos. Sin embargo, la carga computacional impuesta por las técnicas de clasificación de esta capa se ve compensada con una mejora significativa en los resultados de la clasificación.

4.3 Conclusiones

En este capítulo se ha presentado la arquitectura AIDeMaS para la detección de intrusiones en los entornos de las aplicaciones distribuidas. La arquitectura AIDeMaS se ha diseñado en capas, utilizando un enfoque jerárquico para facilitar el modelado de la arquitectura y la estrategia de detección, y lograr una mayor capacidad de recuperación de errores.

AIDeMaS está basada en una arquitectura multi-agente donde distintos tipos de agentes son asignados en cada una de las capas de la arquitectura para ejecutar las tareas del proceso de detección de intrusiones. En concreto, el corazón de la arquitectura se centra en un mecanismo de clasificación en dos etapas. La primera etapa del mecanismo de clasificación realiza una clasificación ligera de las peticiones de usuario sin una demanda significativa de recursos, a diferencia de la segunda etapa que conlleva una mayor carga computacional de tiempo y recursos, pero mejora en la precisión y fiabilidad de los resultados de la clasificación. En ambas etapas del mecanismo de clasificación se utilizan agentes CBR-BDI que incorporan capacidades de aprendizaje y razonamiento para detectar las intrusiones. Los agentes CBR-BDI ejecutan un ciclo CBR para llevar a cabo la clasificación de la petición de usuario utilizando para ello una técnica de clasificación del campo del aprendizaje automático.

Las técnicas y algoritmos del aprendizaje automático se han convertido en un área de constante investigación en el área de la detección de intrusiones. Una característica principal dentro del aprendizaje automático es el paradigma de aprendizaje de los sistemas. A través de la metodología CBR y la utilización de modelos con capacidad de aprendizaje se pueden construir sistemas capaces de aprender y evolucionar a los cambios de comportamientos de los intrusos y los ataques.

A diferencia de las soluciones existentes, la arquitectura AIDeMaS proporciona un conjunto de ventajas que la convierten en un enfoque novedoso en el campo de la seguridad de los entornos de las aplicaciones distribuidas. AIDeMaS se diferencia de las soluciones actuales en la utilización de la tecnología multi-agente, que la dota de capacidades novedosas como el aprendizaje, razonamiento, autonomía y la solución de problemas desde un punto de vista distribuido. En esta misma dirección, AIDeMaS incorpora los sistemas CBR-BDI que han demostrado ser muy eficientes en problemas similares que requieren un alto nivel de aprendizaje y adaptación a los cambios que ocurren en el entorno. En nuestro caso el sistema CBR-BDI permite a la arquitectura una rápida adaptación a los cambios de comportamientos de los ataques, permitiendo reconocer nuevos ataques y variantes en los ataques conocidos, lo que supone un hecho diferencial con respecto a las propuestas existentes. Otra ventaja de la arquitectura AIDeMaS viene dada por la flexibilidad que proporciona para incorporar diferentes técnicas de clasificación del campo del aprendizaje automático para identificar anomalías que son muy difíciles de detectar con las técnicas tradicionales utilizadas en los sistemas IDS. Finalmente, el enfoque jerárquico de la arquitectura aporta varias ventajas tales como un mecanismo de clasificación en capas fácil de entender, una mejor capacidad de recuperación de errores al estructurar un modelo en capas que permite limitar el impacto del error sin comprometer la arquitectura en su totalidad. Otra ventaja del enfoque jerárquico es la escalabilidad. La arquitectura tiene la capacidad de crecer en la medida que crece la demanda de servicios. En resumen, la arquitectura se

presenta como un enfoque robusto e innovador como solución a los problemas de seguridad en los entornos de las aplicaciones distribuidas.

En este capítulo se ha presentado los distintos componentes de la arquitectura. Se ha descrito la estructura interna de los agentes, la arquitectura multi-agente, la forma como se comunican los agentes, etc., además se ha descrito cada una de las etapas del mecanismo de clasificación. En el siguiente capítulo se evaluará la arquitectura AIDeMaS a través de casos de estudios propuestos.

Chapter 5

Case Studies

Introduction

This chapter presents the most relevant case studies where the AIDeMaS architecture has been applied. The chapter focuses on the cases studies related to the detection and prediction of SQL and XML attacks in environments where web services are exploited.

As shown in this chapter, the AIDeMaS architecture has evolved and has been adapted to the needs required in the case studies, adopting the final structure presented in Chapter 4. The first case study focuses on presenting the strategy and mechanism proposed for the detection of SQL injection attacks, while the second case study presents the strategy used to detect XDoS attacks in web services' environments.

The first case study proposes a technique for the detection of SQL injection attacks based on the syntactic analysis of the SQL queries, aimed at extracting the relevant information for the data analysis. The classification mechanism proposes an innovative strategy bases on CBR-BDI systems. The CBR-BDI system classifier agent described in Chapter 4 provides an advanced classification mechanism.

The second case study proposes a classification mechanism that incorporates CBR-BDI agents with a new two-stage-based classifier. The first stage proposes a mechanism for the immediate classification of the attacks, analyzing the headers of the SQL queries and notably reducing the computational resources required for the classification. However, the accuracy in the classification can be improved with a second stage. The second stage increases the complexity of the classification process but improves the reliability of the classification.

The remaining of the chapter describes the case studies and presents the results obtained with the application of the AIDeMaS architecture.

5.1 Case Study Descriptions

This section describes the case studies where the AIDeMaS architecture has been applied. Sub-section 5.2.1 presents the first case study and shows the application of the AIDeMaS architecture to detect SQL injection attacks. Sub-section 5.2.2 presents the application of the AIDeMaS architecture to detect XDoS attacks. For both case studies the structure of the architecture is presented and its adaption to the case studies is detailed.

5.1.1 SQL Injections Attacks

For several years, databases have been a key element of the technology components in organizations. Nevertheless, security is a serious problem for databases and it has become a complex task due to continuous threats and the emergence of new vulnerabilities (Bertino y Sandhu, 2005). In addition, the recent emergence of mobile technologies such as the Personal Digital Assistant (PDA), Smart Phone and laptop computer, as well as greater interconnection of networks across wireless networks, have caused a revolution in the supply of services (Hayat*, et al.*, 2007). This new philosophy of communication allows users to access information anywhere and anytime. The problem of open environments is the complexity of providing full protection. Over the last years, one of the most serious security threats around databases has been the SQL Injection attack (Halfond*, et al.*, 2006). In spite of being a well-known type of attack, the SQL injection remains at the top of the published threat list. The solutions proposed so far seem insufficient to block this type of attack because the vast majority are based on centralized mechanisms (Halfond y Orso, 2005a), (Huang*, et al.*, 2003), (Kosuga*, et al.*, 2007) with little capacity to work in distributed and dynamic environments. Furthermore, the detection and classification mechanisms proposed by these solutions lack the learning and adaptation capabilities for dealing with attacks and variations of the attacks that may appear in the future.

This study presents *SQL-CBR Multi-agent System* (SCMAS), a distributed hierarchical multi-agent architecture for blocking database attacks. SCMAS proposes a novel strategy to block SQL injection attacks through a distributed approach based on the capacities of the SQLCBR agents, which are a particular type of CBR-BDI agent (Bajo*, et al.*, 2008), (Corchado y Laza, 2003). The philosophy of multi-agent systems allows SQL injection attacks to be dealt with from the perspective of the elements of communication, ubiquity and autonomous computation, and from the standpoint of a global distributed

system. Every component in SCMAS interacts and cooperates to achieve a global common goal. SCMAS presents a hierarchical organization structured by levels or layers of agents. This hierarchical structure distributes roles and tasks for the detection and prevention of SQL injection attacks. The agents of each level are assigned specific tasks which they can execute regardless of their physical location.

The SQLCBR agents are CBR-BDI agents (Corchado y Laza, 2003) integrated in the SCMAS architecture; their internal structure and capacities are based on mental aptitude (Georgeff y Lansky, 1987). The use of SQLCBR agents with advanced capabilities for analyzing and predicting SQL attacks is the main feature of the architecture. These agents are characterized by the integration of a CBR mechanism (Case-Based Reasoning) (Aamodt y Plaza, 1994), (Bajo*, et al.*, 2008) in a deliberative BDI Agent. This mechanism provides the agents with a greater level of adaptation and learning capacity, since CBR systems make use of past experiences to solve new problems (Corchado y Laza, 2003). This is very effective for blocking SQL injection attacks as the mechanism uses a strategy based on anomaly detection (Mukkamala*, et al.*, 2005).

The SQLCBR agents designed within the framework of this research are located on the upper levels of the hierarchical architecture. There are two types of CBR-BDI agents that incorporate two novel strategies for both the classification of SQL injection attacks and the prediction of negative behaviors by users. The first type of SQLCBR agent is a classifier agent called "Anomaly", which analyzes SQL queries and then classifies them according to whether the query is defined as attack or not attack to the database. This classifier agent incorporates a novel mechanism in the adaptation stage of his CBR cycle based on a mixture of neural networks. Neural networks are an effective method of classification (Chen*, et al.*, 2009) for problems of this type since current methods such as the Bayesian method (Hernandez, 2007), Exponential Regression model (Martín*, et al.*, 2007), Polynomial Regression Model (Martín*, et al.*, 2007) or Lineal Regression model (Martín*, et al.*, 2007) not only provide solutions more slowly, but solutions that are less effective compared to neural networks. Using a mixture of neural networks we can merge two networks that use neurons with distinct activation functions. As a result of this process, the Classifier agent demonstrates a remarkable improvement in the performance of the classifier since it accounts for the assessment carried out for the two neural networks and avoids conflictive cases that the networks cannot resolve on their own. The second type of SQLCBR agent is the Forecaster agent, which incorporates a mechanism to predict the behavior based on a time series technique. This mechanism uses the history of requests made by a user to predict current behavior.

The aim of this study is to describe the SCMAS architecture and present the preliminary results obtained after the implementation of an initial prototype. These results demonstrate each of the following: the effectiveness of the solution in minimizing and predicting attacks, a higher performance obtained by

distributing the workload among the available nodes in the architecture, a greater capacity for learning and adaptation provided by the SQLCBR agents, and the flexibility to be adapted to many scenarios in which information is susceptible to an SQL injection attack.

The next section presents the SCMAS architecture in greater detail.

5.1.1.1 SCMAS: A Solution based on Multi-agent System

Recent software applications implemented in the electronic commerce, industry and health sectors, among others (Corchado*, et al.*, 2008a), (Bajo*, et al.*, 2006a) are suited to work in dynamic environments with ubiquitous access to information, and require the use of mobile technologies. As such, the problem of providing protection against SQL injection attacks requires a different approach. A solution based on a multi-agent architecture presents the most suitable features for resolving the problem. The present study proposes the use of a distributed and hierarchical architecture depicted in Figure 5.1 to detect and block SQL injection attacks. It is based on a totally innovative approach since there is no known architecture with these characteristics for resolving the problem of SQL injection attacks.

As depicted in Figure 5.1, the distributed resolution of problems balances the workload, facilitates the recovery from error conditions, and also avoids centralized traffic. While requests in current environments are carried out from several devices, it is preferable for specialized agents to monitor requests from various strategic points and avoid having all requests go directly to the database. Similarly, the analysis, classification, and prediction capabilities, among others, are distributed in a layered structure, where the agents that make up the architecture are assigned specific roles to perform their tasks. Moreover, the distribution greatly simplifies the capacity to recover from errors or failures because if an agent fails, it is immediately replaced without affecting the other agents at the same level or in other levels. Additionally, SCMAS architecture uses a model based on a hierarchical model that reduces the complexity of tasks such as monitoring devices and users, classifying user requests, predicting behavior, evaluating the final solution etc. Distributing the functionality at each level, while maintaining each level independent, allows new changes to be easily adapted. Each level of architecture holds a collection of agents with well-defined roles that allow their tasks and responsibilities to be defined. The architecture has been divided into 4 levels so that the specific tasks are assigned according to the degree of complexity. Figure 5.1 illustrates the SCMAS architecture with each level and the respective agents. The details of each type of agent located at the different levels of SCMAS architecture are presented in Table 5.1.

Table 5.1. SCMAS Architecture's agents

Agent types	Architecture Level	Quantity	Abilities / Tasks
Sensor	1	n = number of host	Located in each of the devices with access to the database. They have 3 specific functions: a) Capture datagrams launched by the devices. b) Order TCP fragments to extract the request's SQL string. c) Provide Syntactic analysis of the request's SQL string.
FingerPrint	2	n= workload available	Gets the results supplied by the Sensor agent. Its function includes a pattern matching of known attacks. A database with previously built patterns allows this task.
DBPattern	2	1	Cooperates with the FingerPrint agent in the retrieval of patterns from the database. It is also responsible for updating and adding new patterns to the patterns database.
Anomaly	2	n= workload available	A core component of the architecture, it carries out a classification of SQL strings through detection anomalies. It integrates a case based reasoning (CBR) mechanism. In the reuse stage of the CBR cycle it applies a mixture of neural networks to generate a classification (legal, illegal or suspicious).
Manager	3	1	Responsible for the decision-making, evaluation and coordination of the overall operation of architecture. Evaluates the final decisions of classifications and manages attack alerts and coordinates the actions necessary when an attack is detected. Decisions are based

			on a method of voting among the Anomaly agents.
Forecaster	3	n= workload available	Another core agent in the architecture. Responsible for predicting an attack based on user behavior. Using a technique based on moving averages integrated into the reuse phase of the CRB cycle, the agent is able to predict user behavior and determine an attack against the database.
LogUser	3	1	It is responsible for updating the profile of application users. The requests made to the database are registered, and user profiles are labeled based on the historical behavior. Cooperates with the Anomaly agent to update the users- log.
DB	3	1	It is responsible for executing queries to the database once the requests are classified as legal, and getting the results.
Interface	4	1	It facilitates the interaction between a human expert in charge of security and architecture. It is equipped with the ability to run on mobile devices to achieve direct communication with security personnel whenever an attack is detected. It also facilitates the implementation of adjustments in the setup of the architecture.
Response	4	1	Responsible for formatting and delivering the results of request in device interfaces. The results of valid requests are sent to users; however, invalidated requests are rejected and notified with a warning message.

The complexity of the components of the architecture increases with each level of hierarchy. The entities at each layer use the functionalities that have been provided to them by entities at the bottom layer. The functions have been divided according to the SCMAS levels. The operational tasks are performed at layer 1 and include: capture traffic generated by user requests across different devices, retrieve the SQL string from the capture, and execute a syntax analysis of the SQL string. Layer 2 contains the tactical / strategic tasks for the detection and classification of user requests. The entities at layer 3, the administration layer, carry out tasks related to determining the classification of requests. Additionally, they coordinate the overall functioning of the architecture and determine which measure to take when an attack has been detected. Finally, the tasks of interaction with the user are performed in the top layer of the architecture called the interaction layer, which provides an interface to access services of the architecture.

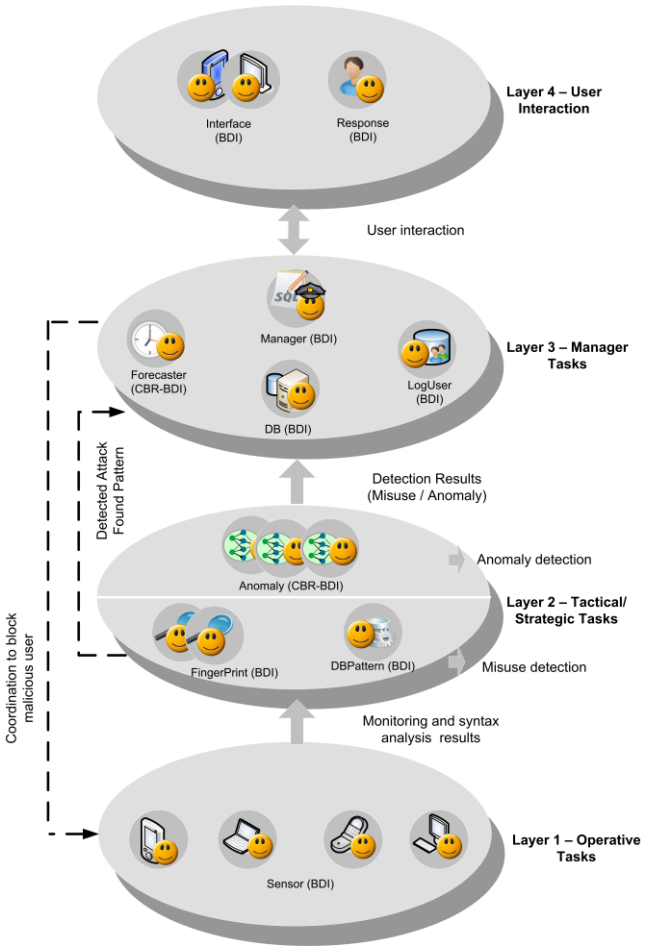

Figure 5.1. SCMAS architecture – Levels and agents

5.1.1.2 Cooperation of agents for task execution

In SCMAS architecture each agent is equipped with the ability and resources to achieve its own goals. However, to achieve the overall goal of the architecture (classifying user requests and predicting the behavior of the hacker) it is necessary to have cooperation and communication among the architecture's agents. This subsection addresses the way in which the types of agents within SCMAS cooperate and collaborate. Different situations in which agents work together to achieve a particular goal are: detection by misuse detection (FingerPrint, Sensor and DBPattern agents cooperate by applying misuse detection to accomplish a task); classification of the user requests (FingerPrint and Anomaly agents cooperate by accomplishing tasks related to the classification of user requests); resolving user requests (Manager, Anomaly, Interface and LogUser cooperate to resolve the task of the classifying user requests); responding to user requests (Manager, DB and Response cooperate to provide an answer to user requests); and behavior analysis and blocking of malicious users (Manager, Sensor and Forecaster agents cooperate in the analysis of user behavior for blocking attacks by hackers). Below, an example of cooperation explains how Misuse detection resolves a detection task.

Table 5.2. Cooperation –Detection based on Misuse detection

Agents	Description
Sensor FingerPrint DBPattern	Sensor agents send their results (transformed SQL string, syntax analysis data, and user data) to a specific FingerPrint agent. The FingerPrint executes a pattern matching with known SQL attack patterns. The FingerPrint agent requests a set of SQL patterns from the DBPattern agent, which manages the pattern database. The DBPattern agent retrieves and sends the set of patterns to the FingerPrint. It is also possible for the DBPattern agent to update the pattern database.

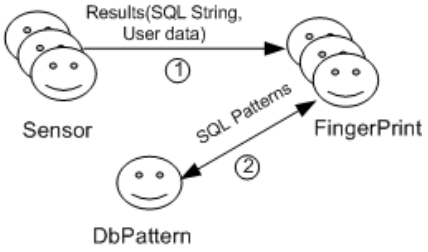

Figure 5.2. Cooperation to resolve the task based on misuse detection.

As shown in Figure 5.2 and Table 5.2, the cooperation to detect attacks based on misuse detection involves the Sensor agent, which belongs to the bottom layer, and the DBPattern and FingerPrint agents, which are located at level 2. Figure 5.2 illustrates how the Sensor agent gets the requested SQL string, analyzes it and sends the results to the FingerPrint. The FingerPrint agent carries out a pattern matching. It requires the DBPattern agent to search the database and retrieve patterns similar to the SQL query being analyzed.

- Communication among agents

In distributed environments where the agents are applied as solutions, it is essential to provide the necessary mechanisms for communication among the agents so they can perform their tasks efficiently. The cooperation among agents is highly dependent on the efficiency of the communication mechanisms which support the interaction. As part of the architecture that is proposed in this study, communication mechanisms were identified at each level of the architecture (intra-layer communication), and a global communication among levels for the execution of tasks (communication inter-layer). The SCMAS architecture considers the use and control of mobile devices, laptop computers and workstations, and takes the use of wireless networks into account.

The SCMAS architecture was developed following the recommendations made by FIPA (Foundation for Intelligent Physical Agents) (FIPA, 2007). The physical transfer of messages is carried out using HTTP (HyperText Transport Protocol) and MTP (Message Transport Protocol) as transport protocols. Another component in communication agents is the communication language. In SCMAS architecture, communication among agents is carried out through the exchange of messages. Messages in SCMAS architecture are based on the standard FIPA CAL, which is based on the speech act theory in which messages are considered communicative acts (Chaib-draa y Dignum, 2002). Figure 5.3 presents a model of messages exchanged between agents in the SCMAS architecture. Figure 5.3(a) presents a FIPA CAL message showing attributes defined and standardized by FIPA. It is important to note that not all attributes are required in a FIPA CAL message. Figure 5.3(b) provides an example of a message relayed between a Sensor agent located on the bottom layer of the architecture and a Fingerprint agent located on level 2. The sensor agent provides the monitoring results by sending the results of the SQL string's syntax analysis and user data. Figure 5.3(c) makes a graphic representation of the message exchanged between Sensor01 agent and FingerPrint01 agent.

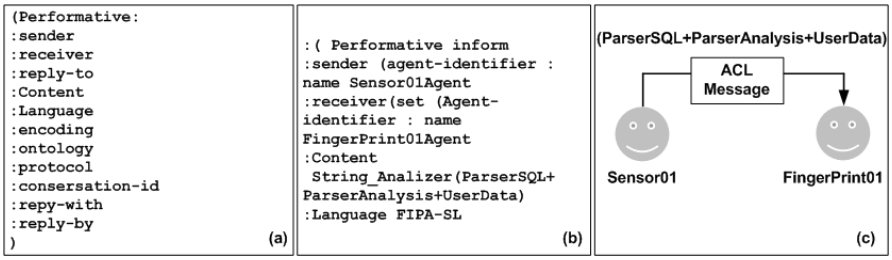

Figure 5.3. Model of the messages exchanged between agents in the SCMAS architecture.

Finally, to achieve interaction and cooperation among agents within the SCMAS architecture, a set of communication protocols were defined. Figure 5.4 presents two examples of the communication between two agents through a protocol diagram. Figure 5.4(a) shows the communication between the FingerPrint agent and the Pattern agent to request SQL patterns stored when the misuse detection is applied. Figure 5.4(b) shows the message sent by the FingerPrint agent when it sends the results generated by the capture of the SQL query, string analysis and user data.

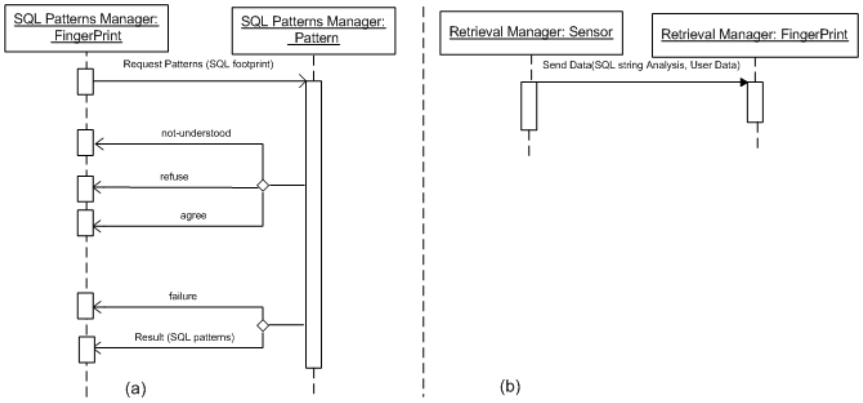

Figure 5.4. Communication pattern during the exchange of a message between agents

As the final step, elements to provide security in agent communication were taken into account. This resource was provided by a secure channel through the HTTPS protocol (Hypertext Transfer Protocol Secure) (Rescorla y Schiffman, 1999). Moreover, the internal-communication among agents of SCMAS architecture was secured by means of JADE-S (JADE Board, 2005), a JADE plug-in that supports user authentication and agents, encryption and message signature.

5.1.1.3 SCMAS Agents with Reasoning Capabilities

The SQLCBR agents proposed in the framework of this research are a CBR-BDI type of agent specially adapted to resolve the SQL injection attack problem. These agents use the concept of CBR to gain autonomy and improve their problem-solving capabilities. The method proposed in (Corchado y Laza, 2003) facilitates the incorporation of case-based reasoning systems as a deliberative mechanism within BDI agents, allowing them to learn and adapt themselves, lending them a greater level of autonomy than what is normally found in a typical BDI architecture (Bratman*, et al.*, 1988). Accordingly, SQLCBR-agents can reason autonomously and therefore adapt themselves to environmental changes. The case-based reasoning system is completely integrated within the agent architecture. The SQLCBR are classifier and predictor agents that incorporate a "formalism" that is easy to implement, in which the reasoning process is based on the concept of intention. Intentions can be seen as cases, which have to be retrieved, reused, revised and retained. A direct relationship between case-based reasoning systems and BDI agents can also be established if the problems are defined in the form of states and actions.

SQLCBR agents implement cases as beliefs, intentions and desires which lead to the resolution of the problem. As described in (Corchado*, et al.*, 2003), (Bajo*, et al.*, 2007), each state of a CBR-BDI agent is considered as a belief, including the objective to be reached. The intentions are plans of actions that the agent has to carry out in order to achieve its objectives, which makes each intention an ordered set of actions. Each change from state to state is made after carrying out an action (the agent remembers the action carried out in the past, when it was in a specified state, and the subsequent result). A desire will be any of the final states reached in the past (if the agent has to deal with a situation that is similar to one from the past, it will try to achieve a result similar to the one previously obtained). The SQLCBR agents, which are explained in detail below, use these concepts to define a case structure for SQL injection problems, and include specific novel mechanisms in the different phases of the CBR cycle to improve the tasks of classification and prediction.

- SQLCBR Agent with Classification Capabilities

The SQLCBR Anomaly agent incorporates a reasoning mechanism that allows it to detect SQL injection attacks. This novel detection technique is supported by a prediction model based on neural networks, which is configured for short-term predictions of intrusions. This mechanism uses a memory of cases

which identifies past experiences with the corresponding indicators that characterize each of the attacks. This study presents a novel classification system that combines the advantages of the CBR systems, such as learning and adaptation, with the predictive capabilities of a mixture of neural networks. SQLCBR agents are particularly suitable to be applied to classification problems in dynamic environments, such as the classification of SQL injection attacks, because they learn from past experiences and adapt to changes. The elements of the SQL query classification problem are represented as follows, using CBR terminology:

- Problem Description: Describes the initial situation (information available) before beginning with the classification process. As shown in Table 5.3, the problem description consists of a case identification; user session and SQL query elements.

- Solution: Describes the actions carried out in order to resolve the problem description. As shown, in Table 5.3, it contains the case identification and the applied solution.

- Final State: Describes the state achieved after the solution has been applied. It can take three possible values: attack when a SQL query is considered as malicious, not attack when the SQL query has been executed without malicious problems, or suspect when the SQL query has been executed but there still exist doubts about the real nature or intentions of the user. The multi-agent architecture incorporates the Manager agent, which allows an expert to evaluate the classification.

Table 5.3. Structure of the problem definition and solution for a SQL query classification

Problem description	Type	Solution	Type
IdCase	Integer	Idcase	Integer
Sesion	Session	Classification_Query	Integer
User	String		
IP_Address	String		
Query_SQL	Query_SQL		
Affected_table	Integer		
Affected_field	Integer		
Command_type	Integer		
Word_GroupBy	Boolean		
Word_Having	Boolean		
Word_OrderBy	Boolean		
Numer_And	Integer		
Numer_Or	Integer		
Number_literals	Integer		
Number_LOL	Integer		

Length_SQL_String	Integer
Start_Time_Execution	Time
End_Time_Execution	Time
Query_Category	Integer

The reasoning mechanism of the Anomaly agent is responsible for classifying the SQL database queries made by the users. When a user makes a new request, it is first checked by a pattern matching mechanism. This mechanism is based on a set of well-known patterns which are stored in a database that handles a significant number of signatures not allowed on the user level, such as symbol combination, binary and hexadecimal encoding, and reserved statement of language (*union*, *execute*, *drop*, *revoke*, *concat*, *length*, *asc*, *chr*, etc.). If the FingerPrint agent detects a known signature using the pattern matching mechanism, the query is automatically identified as an attack and the classification process is complete. In order to identify other SQL attacks that do not match a known pattern, the Anomaly agent uses the CBR mechanism, which must have a memory of cases with the structure described in Table 5.3. The problem description for a case is obtained by a string analysis technique over the SQL query. This process can be easily understood through the following example: Let us suppose that a query with the following syntax: "Select field1, field2, field3 from table1 where field1 = input1 and field2=input2" is received. If the fields input1 and input2 are used to bypass the authentication mechanism with the following input data: "Select field1, field2, field3 from table1 where field1 =" or 9876 = 9876 -- 'and field2="", then, the analysis of the SQL string would generate the result presented in Table 5.4, with the following fields: $\text{Affected\_table}_{(c1)}$, $\text{Affected\_field}_{(c2)}$, $\text{Command\_type}_{(c3)}$, $\text{Word\_GroupBy}_{(c4)}$, $\text{Word\_Having}_{(c5)}$, $\text{Word\_OrderBy}_{(c6)}$, $\text{Numer\_And}_{(c7)}$, $\text{Numer\_Or}_{(c8)}$, $\text{Number\_literals}_{(c9)}$, $\text{Number\_LOL}_{(c10)}$, $\text{Length\_SQL\_String}_{(11)}$, $\text{Query\_Category}_{(12)}$. The fields Command_type and Query_Category have been encoded with the following nomenclature Command_Type: 0=select, 1=insert, 2=update, 3=delete; Query_Category: -1=suspicious, 0=illegal, 1=legal, 2=unassigned.

Table 5.4. SQL String transformed through the string analysis

c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12
1	3	0	0	0	0	1	1	2	1	81	0

When the Anomaly SQLCBR agent receives a new problem description, it initiates a new CBR cycle to achieve a solution. Figure 5.5 shows the algorithm containing the four steps of the CBR cycle:

- Retrieve: The first phase of the CBR cycle consists of recovering past experiences from the memory of cases, specifically those with a problem description similar to the current request. In order to do this, a cosine

similarity-based algorithm is applied, allowing the recovery of those cases which are at least 90% similar to the current problem description.

- Reuse: The recovered cases are then input for the second phase of the CBR cycle, the reuse phase. In the reuse phase the similar cases recovered are used to train the mixture of neural networks. Once the mixture has been trained, it is able to classify the current problem. The result obtained using a mixture of the outputs of the networks provides a balanced response and avoids individual tendencies (always taking into account the weights that determine which of the two networks is more optimal). Because the mixture of neural networks is composed of two different networks, there are some considerations to take into account in the training process: (i) the neural network with neurons based on the sigmoidal function is trained with the recovered cases that were classified as attack or not attack, whereas the neural network with neurons based on hyperbolic function is trained with all the recovered cases (including those of identified as suspicious). (ii) Moreover, a preliminary analysis of correlations is required to determine the number of neurons of the input layer of the neuronal networks. The data used to train the mixture of networks must not be correlated.

  Avoiding correlated data allows network topologies to be reduced. After removing correlations, only the input variables that are not correlated to each other remain, which is reflected in a smaller number of neurons in the input layer, thus lowering the training time (De Paz, 2008). (iii) Additionally, it is necessary to normalize the data (i.e., all data must be values in the interval [0,1]) after deleting correlated cases. If data are not normalized, an overflow is produced in the outputs of neural networks, causing an error. The cause of the error is the use of the hyperbolic tangent and sigmoidal function as activation functions.

- Revise: An expert evaluates the solution proposed for those cases where the mixture of networks generates output values in the interval $[-0.6,-0.4] \cup [0.4,0.6]$. The remaining cases are automatically stored.

- Retain: The system learns from its own experiences from each of the previous phases and updates the database with the solution obtained in the query classification. All cases marked as efficient in the revise phase are stored.

  As shown in Figure 5.5, an essential element in the classification algorithm executed by the Anomaly agent is the mixture of neural networks that is implemented in the reuse stage of the CBR cycle to predict attacks. The mixture uses two neural networks, both of which are Multilayer Perceptrons, but each of the networks uses a different type of activation function. The general idea of the mixture is that each of these networks obtains an individual solution for the problem by following a particular

strategy. The solutions provided are then combined to find the optimal classification. The following paragraphs provide a detailed explanation of the internal structure of the mixture of neural networks, as well as the way it is used to classify SQL queries as attack or not attack. Figure 5.6 illustrates the structure for the mixture of the neural networks

```
Algorithm_Solution_CBR(new_case)  {CBR Algorithm}
Begin
  cases:=Retrieve(new_case)   {case retrieval function}
  assessment:=Reuse(cases[], new_case) {result of the classification}
  decision:=Revise(new_case, assessment) {revision of the classification}
  If decision then  {If decision is accepted}
     Retain(new_case, solution) {update of the memory base}
  End if
End
Algorithm_Retrieve(new_case)  {Retrieve Algorithm}
Begin
   SQL_Query:=Select cases from Tb_Cases Where
   If [User] and [Ip_adress] then   {If User and Ip Adrress is recognized}
     SQL_Query+="user=[user]and IP_Adress=[IP_adress] and Command_type =
     [Command_type] and Affected_table=[Content(Affected_table)]"
   Else
       If [User] then {If only one User is recognized}
         SQL_Query+="User=[User] and Command_type=[Command_type]and
         Affected_table=[Content(Affected_table)]"
         Else
           If [Ip_adress] then {If only one Ip Adress is recognized}
               SQL_Query+="IP_adress=[IP_adress] and Command_type=
               [Command_type] and Affected_table=[Content(Affected_table)]"
           Else  {If user and Ip Adress are not included in the query}
               SQL_Query+="Command_type=[Command_type] and
               Affected_table=[Content(Affected_table)]"
           End If
       End If
   End If
   cases[]:=executeQuery(SQL_Query) {recovering of the cases in database}
   cases[]:=fsimilirity_cosine(cases[]) {cosine similarity-based algorithm}
   cases[]:=fcorrelation(cases[]) {eliminating correlated cases}
End
Algorithm_Reuse(cases[], new_case) {Reuse Algorithm}
Begin
  blenew_case=false
  If description_new_case<>descripcion_previous_case then
     blnew_case=true  {If user or Ip_adress are different of previous case}
  End If
  If blnew_case then   {If is true then training neural network}
    Input:=Retrieve_Input(cases[]) {Retrieval of input}
    Output:=Retrieve_Output(cases[]) {Retrieval of ouput}
    {Training of the neural network}
    error_training:=Training_Neural_Network(Input, Output)
    if (error_training)=low then
       {Classification by mixture of neural network}
       assessment:=Classification_Neural_Network(new_case)
    Else
       Exception(Error_Code, Description) {Imposible to classify new case}
    End If
  Else
   {Classification by mixture of neural network}
    assessment:=Classification_Neural_Network(new_case)
  End If
End
Algorithm_Revise(new_case, assessment)  {Revise Algorithm}
Begin
   Boolean decision
   If new_case=complete and assessment=optimal then {Evaluation by an expert}
      decision:=true
   End If
End
Algorithm_Retain(new_case, solution) {Retain Algorithm}
Begin
   executeUpdate(new_case, solution) {Update case memory with new case}
End
```

Figure 5.5. Algorithm of the Cycle CBR for classifying SQL query

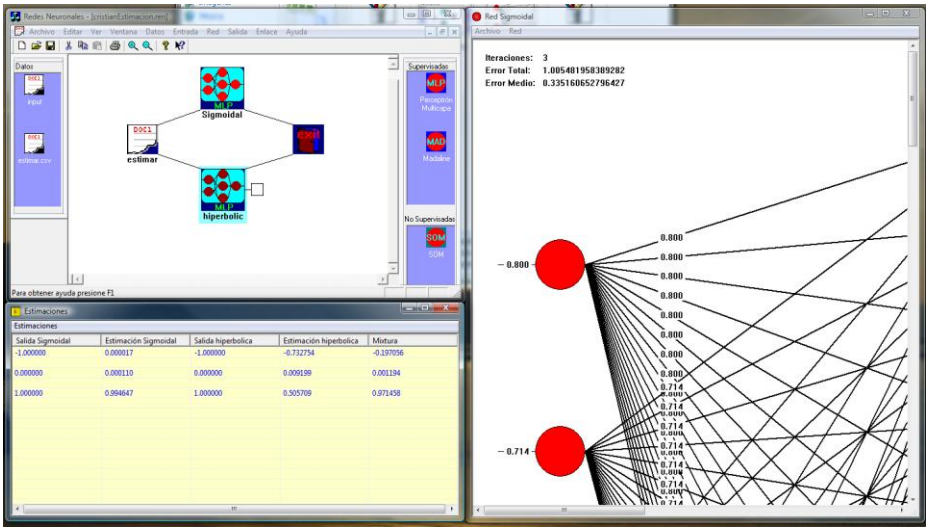

Figure 5.6. Snapshot of the mixture of the neural networks.

As shown in Figure 5.6, the new case is presented to both neural networks at the same time, after which each of the neural networks will give its own decision regarding the classification. The neural network based on a sigmoidal function gives two results (illegal or legal) and the neural network that uses a hyperbolic tangential function produces three results (illegal, legal or suspicious). The learning algorithm for the neural networks and an explanation for the difference in each type of network are detailed in (De Paz, 2008). Because the mixture is composed of two Multilayer Perceptrons that use different activation functions, the learning algorithm has been particularized by taking both possible activation functions into account (De Paz, 2008).

- The Sigmoidal activation function has its range of possible values at the interval [0,1]. It is used to detect if the request is classified as an attack or not attack. The value 0 represents an illegal request (attack) and 1 a legal request (non attack). The Sigmoidal activation function is the most commonly used activation function for classifications between two groups. This function has the drawback of only placing classifications in two groups, that is:

$$f(x) = \frac{1}{1+e^{-ax}} \quad (1)$$

Where the value *a=1* is used in the equation.

- The hyperbolic tangential function has its range of possible values in the interval [-1,1]. It is used to detect if the request is an attack, not attack or suspicious. The hyperbolic tangential function allows more possible cases than the sigmoidal function. The value 0 represents an illegal request, value 1 represents a legal request, and value -1 are suspicious requests. The hyperbolic tangential function is suitable for classifying the request into three groups. The hyperbolic tangential activation function is given by

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad (2)$$

If only one network with a sigmoidal activation function is used, then the result provided by the network would tend to be attack or not attack, and no suspicious results would be detected. On the other hand, if only one network with a hyperbolic tangent activation is used, then a potential problem could exist in which the majority of the results would be identified as suspicious although they were clearly attack or not attack. The mixture provides a more efficient configuration of the networks since the global result is determined by merging two filters. This way, if the two networks classify the user request as an attack, so too will the mixture; and if both agree that it is not an attack, so will the mixture as well. If there is no concurrence, the system uses the result of the network with the least error in the training process or classifies it as a suspicious. In the reuse phase the two networks are trained by a back-propagation algorithm for the same set of training patterns (these neural networks are named Multilayer Perceptron), using a Sigmoidal activation function (which will take values in [0,1], where 0 = Illegal and 1 = legal) for a Multilayer Perceptron and a hyperbolic tangent activation function for the other Multilayer Perceptron (which take values in [-1,1], where -1 = Suspect, 0 = illegal and 1 = legal).

The response of both networks is combined obtaining the mixture of networks. The networks that are combined should have the same number of neurons in output and input layers. In addition, it is necessary that the training set used in both networks be the same (same output and input data, and number of patterns), since the intention is to assess which of the two networks learns best from the training set. However, the number of neurons in the hidden layer can be variable. The steps required to mix the outputs of the neural networks are:

- Determine the best network for the output neuron (the network that provides a minimum average absolute error for the output neuron which is denoted by $E.A.M.S^{r}$, where r indicates the network considered).

- Consider outputs of each network according to the good results obtained during the learning process.

To formalize the mix of networks, a new counter r is added to indicate the number of the network being considered.

$$E.A.M.S^r = \frac{1}{2q}\sum_{p=1}^{q}|Y^{pr} - d^{pr}|, r = 1,2; \quad (3)$$

Where Y^{pr} is the output obtained by training pattern p of the network r, d^{pr} is the desired output according to training pattern p of the network r, and q the number of training patterns. The networks are sorted from best to worst performance, which is valued through the rate of learning, that is, of least to greatest $E.A.M.S^r$. Once sorted, the output obtained from merging both networks is calculated, resulting in a weighted function of the output from each network. The fact that the network has provided a better learning rate must be taken into account and positively weighted. The output obtained from the mixture of neural networks with output k for the Z networks is denoted as Y^2, as is presented in (4), where $a = 1$ if the networks are previously sorted from best to worst.

$$y^2 = \frac{1}{\sum_{r=1}^{2} e^{|-a-r|}} \sum_{r=1}^{2} e^{|-a-r|}\, y^r \quad (4)$$

Compared to traditional techniques, neural networks are a good alternative for classification problems as evidenced by the empirical results presented in the results section. The reason for choosing a mixture of networks in classifying SQL injection attacks is that the use of neural networks alone is limited when it comes to making decisions. As a clear example of this type of problem, suppose that we approach the problem using a single neural network, for example one based on neurons with a sigmoidal activation function. In this case, the network could not conclude anything automatically and the intervention by a human expert would be required. However, if instead of considering just one network, we use a mixture of networks, the system is capable of solving this type of situation. Table 5.5 presents possible situations for the output of the neural network with a sigmoidal function and a value of 0.5. This network has the least error training of the two systems, and different cases of output are evaluated by the network with tangential hyperbolic activation function.

Table 5.5. Possible situations where the mixture of neural networks would not generate a solution.

Sigmoidal Network Output	Hyperbolic Network Output	Mixtured Output
0.5	0.5	(1/(1+Exp[1]))(0.5+0.5*Exp[1])=0.5
0.5	-0.5	(1/(1+Exp[1]))(0.5-0.5*Exp[1])= -0.231059
0.5	0.25	(1/(1+Exp[1]))(0.5+0.25*Exp[1])= 0.317235
0.5	-0.25	(1/(1+Exp[1]))(0.5-0.25*Exp[1])= -0.0482939

0.5	0.75	(1/(1+Exp[1]))(0.5+0.75*Exp[1])= 0.682765
0.5	-0.75	(1/(1+Exp[1]))(0.5-0.75*Exp[1])= -0.413823

As shown in Table 5.5, if the mixture is used, the only situation requiring intervention by a human expert is the one in which both networks give an output value of 0.5. The likelihood of this happening is $\frac{1}{11}$; applying the Laplace rule which tells us that the probability of the occurrence of an event is the quotient of favorable cases and possible cases. When working with one digit decimal numbers, there is one case in which the mixture cannot decide: when the value from among the 11 possible interval values [0,1] is equal to 0.5. The probability that a network with hyperbolic activation function will take on a value of 0.5 is $\frac{1}{21}$ (there continues to be one case with one digit decimal numbers in which the mixture cannot decide from among the 21 possible values interval [-1, 1]). So, the likelihood of needing the intervention by a human expert by using a mixture of the two networks, as compared to both networks individually would be: $\frac{1}{11} * \frac{1}{21} = \frac{1}{231} = 0.0043$

- SQLCBR Agent with Prediction Capabilities

This is a SQLCBR agent specially designed to predict the behavior of suspicious users. When a user makes requests that are seen as malicious or suspicious, it is very important for the system to have a mechanism to predict when and what type of request the user is going to make next. Such predictions allow the system to anticipate the actions of potentially dangerous users and block the attacks before they occur. The use of CBR mechanisms is especially suitable for this type of problem since it is possible to use past experiences to predict future behavior. This study proposes a strategy of a CBR cycle based on the use of Time Series at the reuse stage. This is a novel system for predicting behaviors that will improve the results that until now were only obtained with other existing methods. The reason is that each user identified as a hacker usually follows unique patterns of action, especially if he or she has already been successful with previous attacks. In other words, there is a high component of similarity with past experiences. Among the techniques that are currently used to predict the behavior that a hacker can engage in when executing SQL injection attacks are the Linear Regression Method (Martín*, et al.*, 2007), Decision Trees (Quinlan, 1986) and the Time Series (Martín*, et al.*, 2007). Table 5.6 shows the efficiency of these methods based on 100 cases of SQL injection attacks. As seen in Table 5.6, the prediction based on Time Series and the predictions based on Decision Trees provide excellent results, whereas the results of the Linear Regression method show a much lower degree of success.

Table 5.6. Effectiveness of the prediction techniques

Method	Efficient (%)
Linear Regression	50
Tree Decision	95
Temporal Series	97

The CBR-BDI mechanism based on a time series analysis is responsible for predicting behaviour according to the requests made by users. When a user makes a new request, it is matched against well-known patterns of attack. If there is a match, the request is automatically identified as an attack. Additionally, the request is identified as an attack by the Forecaster agent in cases where there are several unsuccessful attempts at certain requests, which is known as a brute force attack. In order to identify the rest of the SQL attacks, the system uses CBR, which must have a memory of cases dating back at least 4 weeks, and store the following variables: User ID, Host ID & IP Address, Request, Completion (results of user requests, which returns a zero if successful, otherwise it returns the error code describing the reason for the failure), Valid Time (which has two values, startup time and time of completion). The structure of a case for the behavior prediction problem of users is shown in Table 5.7.

Table 5.7. Structure of a case for user behavior prediction problem

Problem Description	Solution: Prediction
User's log-queries	Next request to be made by the user
User profile	Estimated time to resolve the request.
Current query (Case newly classified)	

As formerly indicated, in order to carry out a suitable prediction it is necessary to have previously stored the variables that make up the problem description during a period of at least 4 weeks. With these data stored, a multivariate time series is created and certain trends are eliminated (overall direction of the variable in the observation period), using the moving averages method. This method involves the replacement of one series by another series formed with the means of several values from the original series. The moving averages method of size *n* is used when each of the means of the new series created is equal to the means of *n* elements adjacent to the series. So, for example: $m(x_t) = \frac{x_{t-1}+x_t+x_{t+1}}{3}$, is the moving average for three points. If *n* is odd then only the first series of means is calculated. $m(x_t) = \frac{x_{t-1}+x_t+x_{t+1}}{3}$ is the moving average for three points. If *n* is even, then the second series of means is collected.

In other words, $m(x_t) = \frac{\left(x_{t-2}/2\right)+x_{t-1}+x_t+x_{t+1}+\left(x_{t+2}/2\right)}{4}$, is the moving average for four points.

When a user executes a request, the multivariate time series informs us about what type of action to take, what it expects the following requests will be, and what the approximate time of the next action taken will be. This will allow us to take the necessary measures and anticipate movements made by the user. The basic idea in obtaining the time series is to take into account both the exact moment at which requests are made by users, as well as earlier requests made. Each new request that is executed has a direct correlation with the previous request, as well as with the response given by the system. Thus, the request made at the exact time *t +1*, depends on the request made at the exact time *t* and of the success or failure of the request made in time *t*. The time series takes the patterns stored in the database into account and applies the moving averages method. The time series labels the axis of the ordinate with the number of requests, the request related to time *t*, and the success or failure obtained with such request. To predict the next request, the series with the actions carried out by the user until that instant of time are recovered. Then a search is executed on the stored data in order to obtain the sequence of actions most likely to be carried out by the user. In this way, the time series can be represented as a function: Request, Completion (results of user requests, which return a zero if successful, otherwise an error code describing the failure is returned.) and Valid Time (which has two values, startup time and time of completion).

5.1.1.4 Practical Application of the Architecture: A Medical Database

A case study was proposed to test the effectiveness of a SCMAS prototype. The prototype was evaluated by a previously developed multi-agent system installed in a geriatric facility/care home (Corchado*, et al.*, 2008a). The implemented multi-agent system has improved the security of the patients, facilitated care-giver activity and guaranteed an adequate level of efficiency. The system was developed in a distributed environment containing devices such as PDA, notebook computers and wireless internet access. A back-end database stores and supplies information. The database manager is MySQL. The participants, including nurses, doctors, patients, social workers and other employees can be seen in Figure 5.7. The medical staff in charge of patient care was comprised of 2 doctors, 10 nurses and 1 social worker. Thirty patients were being observed and attended to by the multi-agent system. Each nurse was equipped with a PDA, so a total of 10 PDAs were executing queries on the database during the work day. With these data, we prepared an attack scenario.

The equipment necessary for performing the test included 2 workstations and 3 PDAs. The test was carried out over a period of 30 uninterrupted working days.

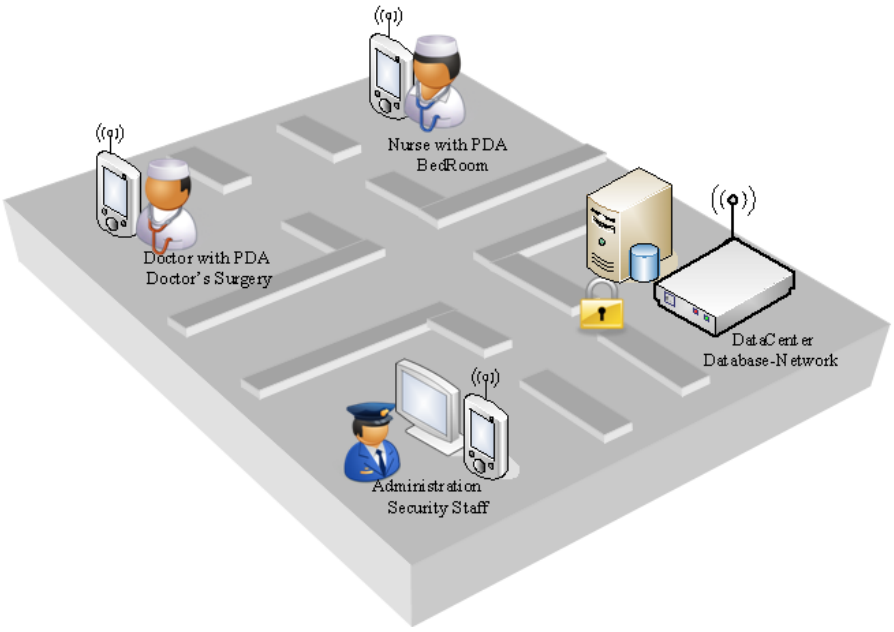

Figure 5.7. Abstract scenario of the real environment (Geriatric Care Home)

During the execution of the multi-agent system, several types of SQL queries were carried out on the database. The queries were related to patient treatments, scheduling the work day for the nurses, etc. Most of the queries were executed from PDAs. The PDAs are used by doctors and nurses to accomplish their tasks. To facilitate the evaluation of the prototype, we focused on the nurse role. A main volume of queries was generated each time a plan was assigned to a nurse. The plans changed for different reasons during their execution and these changes increased the number of queries on the database. When nurses start and finish a task, they send a response through a SQL query. The nurses have direct access to the database system through the application interface on their PDAs. The strategy followed in the case study was based on the execution of queries crafted from 2 attack PDAs. These PDAs were installed with a user interface similar to the nurses' PDAs, but the two attack PDAs are capable of executing tainted queries. When a query is executed from the attack PDA, it carries out a type of SQL injection that has to be captured, analyzed and classified as legal, illegal or suspicious according to our parameters. The FingerPrint agents and Anomaly SQLCBR agents were distributed in the 2 workstations. Because the test was carried out on a real medical database, a special mechanism was built to guarantee the integrity of the database. All the queries executed both by the

nurses' PDAs and the attack PDAs were examined and classified. The test was conducted with a total of 12 PDAs, 10 PDAs assigned to the active nurses and 2 PDAs to execute attacks, and a total of 10,200 queries were sent to the medical database. Each nurse's PDA executed approximately 30 daily queries, and during the 30 days of the test, 9,000 legal queries were carried out. Each of the two attack PDAs executed 20 illegal queries daily. These PDAs sent 40 attacks during a work day. Throughout the 30 day test period, a total of 1,200 attacks targeted the medical database. The volume of queries during the test period allowed us to build a case memory to validate the proposed strategy.

The following example explains the classification process for the SQL queries in SCMAS. Let us assume that a nurse uses her personal identification number and password to log into the system from a PDA, whereby the following SQL string is used to place the request:

SELECT IdNurse, FirstName, SecondName, BirthDate, Age, Sex, SInsurance, salary, BAccount FROM TBNurses WHERE strIdNurse='PA0012'AND strPassword= 'ONeil15'

The string is attacked and the resulting SQL code is:

SELECT strIdNurse, FirstName, SecondName, BirthDate, Age, Sex, SInsurance, salary, BAccount from TBNurses where strIdNurse='' OR 'abcd'='abcd' --' AND StrPassword=''

Let us assume that the first filter applied by the FingerPrint agent did not detect a malicious code and the query continues to the Anomaly agent for its classification in the next layer. Applying a syntactic analysis on the text string for the SQL query, the following values, as listed in the Table 5.8, would be generated.

Table 5.8. Values obtained from the syntactic analysis on the SQL request string

Fields	Values	IdField
Affected_table	1	c1
Affected_field	9	c2
Command_type	0	c3
Word_GroupBy	0	c4
Word_Having	0	c5
Word_OrderBy	0	c6
Numer_And	1	c7
Numer_Or	1	c8
Number_literals	4	c9
Number_LOL	1	c10
Length_SQL_String	165	c11

Query_Category	2	c12

Following the procedure, the correlation data are eliminated and the initial data from the neural network are normalized within the range [0,1]. Table 5.9 presents the results of the normalization of the values for the new case.

Table 5.9. Description of the new normalized case

c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12
0	0	0	0	0	0	0	1	0,4	0,5	0,036	0

To complete this process, the Anomaly agent executes a CBR cycle in order to classify the new SQL string.

- A cosine similarity algorithm is applied in order to recover the cases that have a 90% similarity rate to the new case. Table 5.10 lists the similar cases (SQL strings) that were recovered from the memory of cases.

Table 5.10. Cases (SQL strings) similar to the new case recovered from the memory of cases

SELECT strIdNurse, FirstName, SecondName, BirthDate, Age, Sex, SInsurance, salary, BAccount FROM TBPatients where strIdNurse='' OR 1=1 --' AND StrPassword=''
SELECT strIdNurse, FirstName, SecondName, BirthDate, Age, Sex, SInsurance, salary, BAccount FROM TBPatients where strIdNurse='mysql.user' --' AND StrPassword=''
SELECT strIdNurse, FirstName, SecondName, BirthDate, Age, Sex, SInsurance, salary, BAccount FROM TBPatients where strIdNurse='' OR 2=2 --' AND StrPassword=''
SELECT strIdNurse, FirstName, SecondName, BirthDate, Age, Sex, SInsurance, salary, BAccount FROM TBPatients where strIdNurse='' OR 1=2 --' AND StrPassword=''
SELECT strIdNurse, FirstName, SecondName, BirthDate, Age, Sex, SInsurance, salary, BAccount FROM TBPatients where strIdNurse='' OR '111'='111' '-- AND StrPassword=''

The description of the cases that have been recovered and normalized with the corresponding similarity measure is shown in Table 5.11. The results fall within the range [0,1] where the larger the value, the more similar the recovered case is to the new case.

Table 5.11. Description of cases recovered and normalized with the similarity measure corresponding to the new case.

c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	Measure of Similarity
0	0	0	0	0	0	0	1	0,4	0,5	0,036	0	0,858
0	0	0	0	0	0	0,5	0	0	0	0,072	0	0,370
0	0	0	0	0	0	0,5	1	0,4	0,5	0,036	0	0,930
0	0	0	0	0	0	0,5	1	0,4	0,5	0,036	0	0,930
0	0	0	0	0	0	0,5	1	0,4	0,5	0,133	0	0,933

- In the second phase of the CBR cycle both of the combined networks are trained with cases that were recovered from the first phase of the CBR cycle. Table 5.12 shows the results obtained during the training period for the 5 most similar cases, and the estimate for the new case that has been classified.

Table 5.12. Classification of the mixture of neural networks and the estimate for the new case.

Case	Results
1	0,231
2	0,298
3	0,211
4	0,223
5	0,214
New case	0,278

- The solution is evaluated in the third phase of the CBR cycle. The exit value for the new case (0.278) indicates that the SQL string from the user request is classified as an attack and does not require evaluation by an expert since it falls outside of the given interval.

- Finally, an expert assigns an efficiency rate of 98% in the learning phase and the case is stored in the memory of cases to be used in future situations.

Figure 5.8 provides a graphical representation of the CBR phase, showing the entry and exit data and the tasks that were completed in each phase.

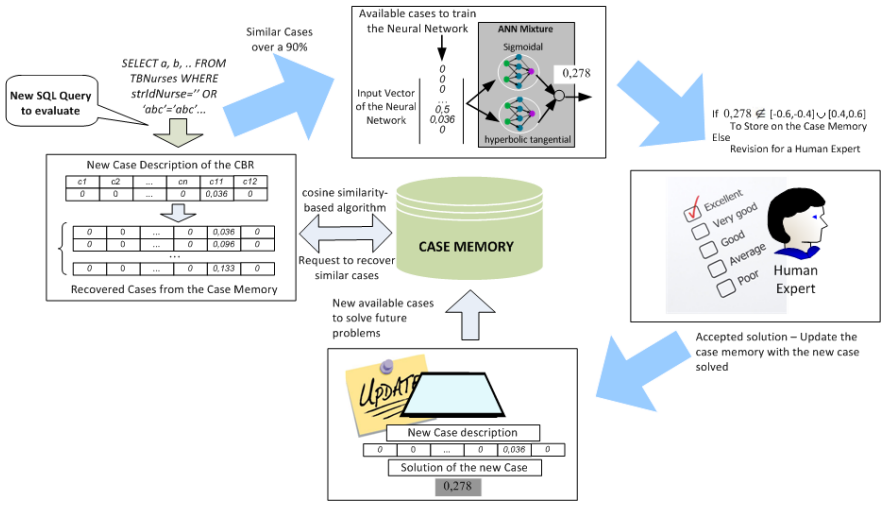

Figure 5.8. Representation of the CBR cycle incorporated within the Anomaly agent.

The next section presents the results obtained after executing the test.

5.1.1.5 Results and Conclusion of the Case Study

The problem of SQL injection attacks on databases supposes a serious threat against information systems. This study has presented a novel solution based on a new hierarchical multi-agent architecture for detecting SQL injection attacks. This solution combines the advantages of multi-agent systems, such as autonomy and distributed problem solving, with the adaptation and learning capabilities of CBR systems. Because current approaches are based on centralized strategies (Halfond y Orso, 2005a), (Huang, *et al.*, 2003), (Kosuga, *et al.*, 2007), the architecture proposed in this study offers a novel perspective in the detection and prediction of SQL injection attacks. The SCMAS architecture provides a hierarchical, distributed structure, which allows a more efficient balance and distribution of the tasks involved in the problem of detecting, classifying, blocking and predicting malicious SQL injection attacks on databases. In addition, this study has presented two interesting mechanisms for improving the classifications of SQL attacks and the prediction of attacks from malicious users. Both mechanisms were implemented through SQLCBR agents, a special type of CBR-BDI agent (Corchado y Laza, 2003) which demonstrates a great capacity for learning and adaptation. The SQLCBR Anomaly agent is a classifier agent that, based on the philosophy of the case-based reasoning mechanisms (Corchado y

Laza, 2003), proposes a new strategy that uses past experiences to classify SQL injection attacks. This strategy differs in its conception from other current strategies and, moreover, incorporates the prediction capabilities that characterize neural networks. The Anomaly agent integrates an innovative model into its CBR cycle consisting of a mixture of neural networks that provides a significant reduction in the error rate during the classification of attacks, and improves the efficiency of the current methods. The second type of SQLCBR agent is the Forecaster agent, which incorporates a mechanism based on time series analysis into the reuse stage of its CBR cycle in order to predict the behaviour of the malicious users, thus allowing preventive actions to minimize possible attacks on the database.

To check the validity of the proposed model, we elaborated a series of tests which were executed on a memory of cases, specifically developed for these tests, which generated attack consults. The results obtained are promising, improving in many cases those obtained with other current techniques, which allows us to conclude that SCMAS can be considered a good alternative for the detection and prediction of SQL injection attacks. The tests were conducted in the following way: first, we evaluated the efficiency of the classification and prediction methods proposed in this research and compared the results obtained to alternative techniques. Then, we evaluated the global efficiency of the architecture by comparing different meaningful parameters before and after the implementation of the system in the test environment. The following paragraphs describe the experiments and discuss the conclusions obtained.

The classification system integrated within the Anomaly agent provided the results shown in Table 5.13, which are promising: it is possible to observe different techniques for predicting attacks at the database layer and the errors associated with misclassifications. All the techniques presented in Table 5.13 have been applied under similar conditions to the same set of cases, taking the same problem into account in order to obtain a new case common to all the methods. Note that the technique proposed in this article provides the best results, with an error in only 0.537% of the cases.

Table 5.13. Results after testing different classification techniques

Forecasting Techniques	Success (%)	Approximated Time (secs)
Anomaly Agent (mixture NN)	99.5	2
Back-Propagation Neural Networks	99.2	2
Bayesian Forecasting Method	98.2	11
Exponential Regression	97.8	9
Polynomial Regression	97.7	8

Linear Regression	97.6	5

As shown in Table 5.13, the Bayesian method is the most accurate statistical method since it is based on the likelihood of the events observed. It has the disadvantage of determining the initial parameters of the algorithm, although it is the fastest of the statistical methods. After taking the errors obtained with the different methods into account, the regression models follow both the neural networks and the Bayesian methods. Because of the non linear behaviour of the hackers, linear regression offers the worst results, followed by the polynomial and exponential regression methods. This can be explained by looking at hacker behaviour: as hackers break security measures, the time they have for obtaining information via their attacks decreases exponentially. The empirical results show that the best methods are those that involve the use of neural networks, and if we consider a mixture of two neural networks, the predictions are notably improved. These methods are more accurate than statistical methods for detecting attacks to databases because the behaviour of the hacker is not linear, but dynamic and chaotic.

The advantage of using a mixture of neural networks not only improves performance provided by other classification techniques, but also improves performance that only neural networks can provide. The mixture has the advantage of reducing the number of cases in which the classifier agent cannot make decisions, thus requiring human intervention in only a few cases. We were able to check the decision of the mixture of networks with that of the human expert for those cases in which a single network did not decide, and found that both the mixture of networks and the human expert were in agreement in 99% of the cases. Figure 5.9 shows the effectiveness in the classification for two individual networks with a distinct activation function, and the effectiveness of the mixture of networks.

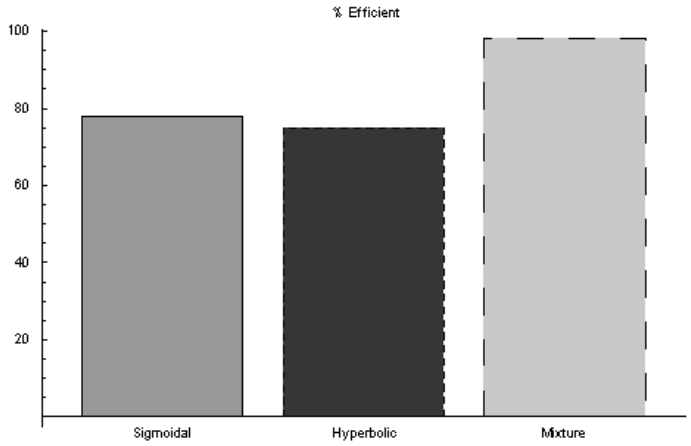

Figure 5.9. Effectiveness in the individual classification of networks and the mixture of networks.

The prediction mechanism integrated within the Forecaster agent also provided good results, as can be seen in Table 5.14. Looking at the behaviour of the time series analysis, we have observed that when the number of training patterns for the neural network increases, prediction error decreases. It is important to note that the number of training patterns is the result of applying filters such as the similarity-based algorithm and the correlation function. These filters meaningfully reduce the quantity of cases and allow improved performance during the training stage. The graph in Figure 5.10 indicates the success of the predictions with regards to the number of training patterns presented in Table 5.14.

Table 5.14. Successful (%) depending on number of training patterns

Number of patterns training	Successful (%)
1000	99.5
900	99.1
700	98.5
500	98.6
300	96.8
100	89

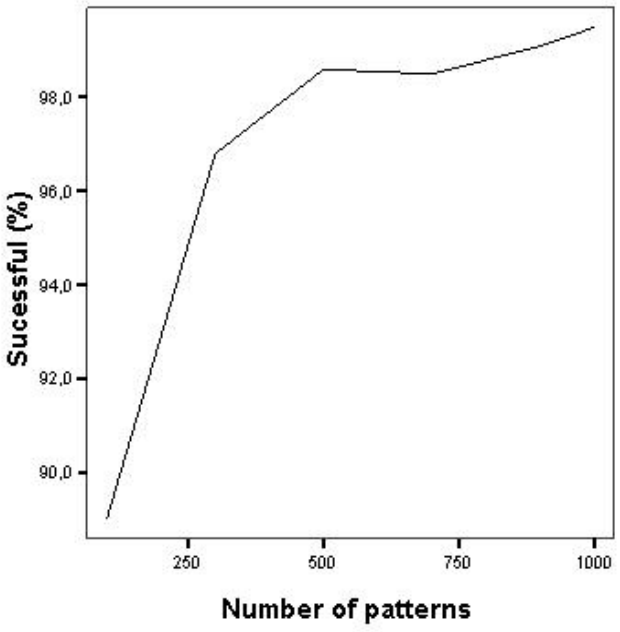

Figure 5.10. Sucessful (%) vs. Number of patterns

Figure 5.10 shows the percentage of predictions with regards to the number of patterns in the training phase. It is clear that with a large number of training patterns the percentage of successful predictions improves. Since we are working with CBR systems, which depend on large amounts of data stored in the memory of cases for each user, the percentage of successful predictions increases, as demonstrated in Figure 5.10. CBR systems need initial information (past experiences) to generalize efficient results. In addition, the time series analyses also need the data that is passed on in order to allow reliable predictions. Figure 5.11 shows that a period of 4 weeks implies an acceptable threshold to carry out predictions with a high rate of success.

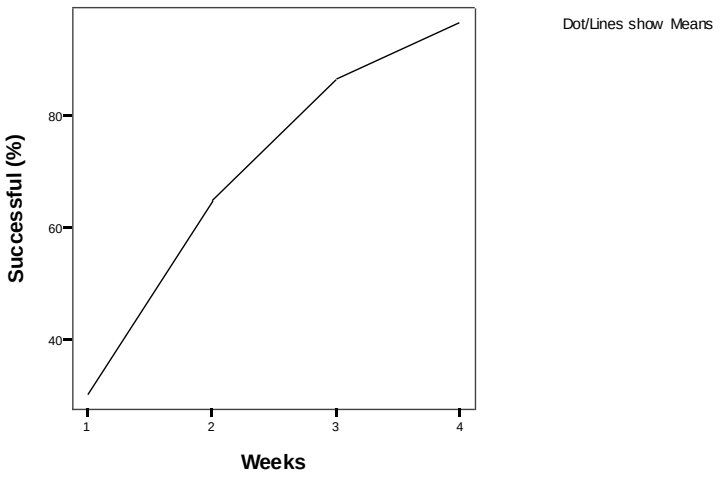

Figure 5.11. Success prediction according to the number of weeks from the database

In order to analyze the impact of multi-agent architecture proposed in the context of this investigation, we have evaluated the percentage of attacks detected before and after implementing the system, as shown in Figure 5.12. Figure 5.12 shows a progressive increase in the percentage of attacks detected successfully over time. As expected, the system does not initially provide a high detection rate, since it works with synthetic data. But as time passes, the system learns and adapts to the environment in which it is located. Thus, with four weeks of stored data, it is possible to obtain an attack detection rate above 87%.

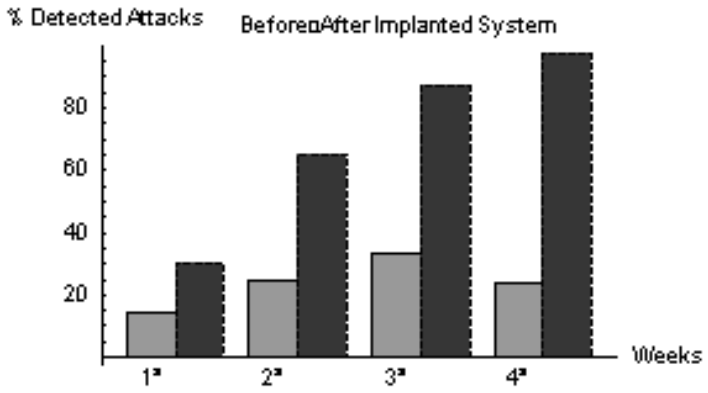

Figure 5.12. Progressive increase in the detection of attacks based on time.

Another meaningful indicator for evaluating the classification system of SQL injection attacks is the percentage of false positives identified by the system. This is an important parameter, because classifying queries as attacks when they are not so can cause serious delays in the system and discomfort to users. The percentages of false positives caused before and after implementing the system are shown in Figure 5.13. As shown, the SCMAS architecture has adapted itself gradually over time. The results obtained during the first three weeks exceeded the percentage of error obtained without the implementation of the system. However, after the fourth week the SCMAS system learns and reduces the rate of false positives.

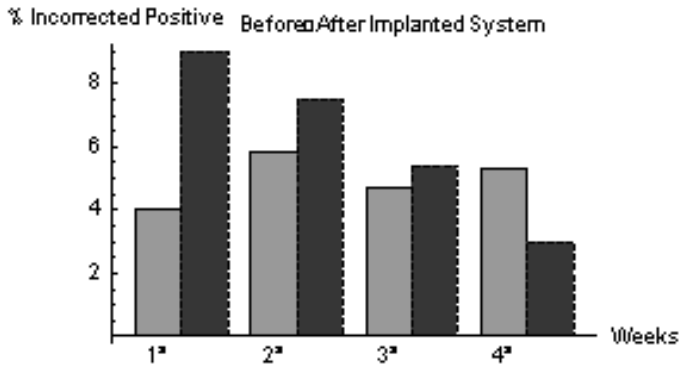

Figure 5.13. Percentage of false positives caused before and after the introduction of SCMAS.

As mentioned, user requests can be classified as attack, not attack or suspicious. The classification of suspected attacks reduces the success rate of the system and requires intervention by a human expert. That is why we have evaluated the percentage of suspicious requests before and after implementing the system. Figure 5.14 shows the percentage of suspicious requests identified by SCMAS compared to those identified by a human expert. As shown in Figure 5.14, the SCMAS system provides rates of 36.8% initially, but this error rate decreases quickly, falling to 4.3% in the fourth week of operation.

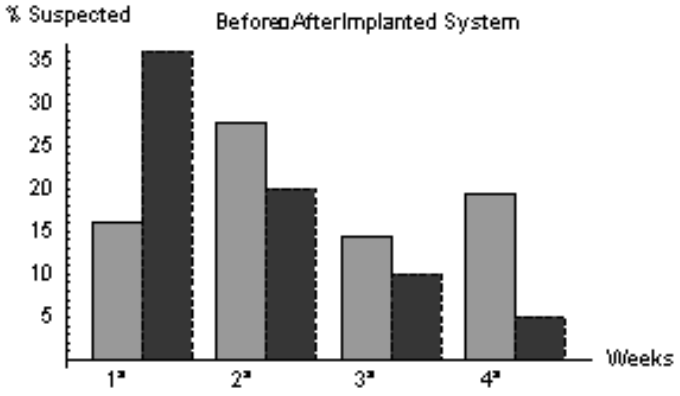

Figure 5.14. Percentage of suspicious requests before and after implemented SCMAS system.

The architecture presented in this study provides a novel strategy for detecting SQL injection attacks. The results are promising and allow us to conclude that the SCMAS architecture considerably improves results provided by current technologies. SQLCBR agents are well-suited to the prediction and classification tasks, and improve current techniques.

5.1.2 Denial of Service Attacks based on XML - XDoS

Web Services have become increasingly relevant not only within private networks for companies and organizations, but also at the level of inter-communication. This trend to inter-communication in the Web Services has made the security a key element in open architectures. However, basic Web Service specifications themselves do not address any security topics. Several additional specifications as WS-Security (Lawrence*, et al.*, 2004), WS-SecurityPolicy (Della-Libera*, et al.*, 2005), WS-Trust (Anderson*, et al.*, 2004b),

WS-SecureConversation (Anderson, *et al.*, 2004a), etc. for Web services security exists, but all these standards focus on the aspects of message integrity and confidentiality and user authentication and authorization. Then, it is necessary to investigate in novel method to protect the servers from denial of services attacks (XDoS), which cause malicious or altered Web services, and affect the availability of the Web services (Gruschka y Luttenberger, 2006). XDoS attacks are due to the fact that XML messages must be parsed in the server, which opens the possibility of an attack if the messages themselves are not well structured or if they include some type of malicious code. Resources available in the server (memory and CPU cycles) of the provider can be drastically reduced or exhausted while a malicious SOAP message is being parsed. A XDoS attack is successfully carried out when it manages to severely compromise legitimate user access to services and resources.

Some approaches focus on preventing XDoS attacks to Web Services architectures (Im y Song, 2005), (Bebawy, *et al.*, 2005), (Wang, 2006), (Loh, *et al.*, 2006), (Gruschka y Luttenberger, 2006), (Padmanabhuni, *et al.*, 2006), (Yee, *et al.*, 2007), (Srivatsa, *et al.*, 2008), (Ye, 2008), (Chonka, *et al.*, 2009), but present as main disadvantage their low capacity to adapt themselves to the changes in the patterns of attack, which causes a reduction of the effectiveness of these methods when slight variations in the behaviours of the known attacks happen or when new attacks appear. Moreover, most of the existing approaches are based on a centralized perspective, which provides facilities for XDoS attacks. In this sense, and focusing on performance aspects, centralized approaches can become a bottleneck when the security is breached, causing a reduction of the overall performance of the application. Another approaches focus on providing solutions to XDoS attacks in Web services environments with a perspective similar to traditional layer 2-4 firewalls and application level firewalls which no longer viewed as an effective way for providing a solution to the Web services. The use of Web services over HTTP makes it hard to use traditional layer 2-4 firewalls to block malicious web services traffic (Bebawy, *et al.*, 2005).

Taking into account the limitations of the existing approaches and the particularities of the new trends in XDoS attacks, this study presents an Adaptive Hierarchical Distributed Multi-agent Architecture (S-MAS) for dealing with XDoS attacks in Web Service environments. The proposed architecture is based in our previous research in SQL injection attacks (Bajo, *et al.*, 2010), (Pinzón, *et al.*, 2008a) where a multi-agent architecture was developed. In this way, some resources are reused and the knowledge acquired in this previous work is adapted to get an evolution of the architecture. This previous architecture has a four-tiered hierarchical design that is better capable of task distribution and error recovery. The classification mechanism integrated within the multi-agent architecture has also evolved, incorporating a two-phase strategy to classify SOAP messages. The first phase applies the initial filter for detecting simple attacks without requiring an excessive amount of resources. The second phase involves a more complex process which ends up using a significantly higher

amount of resources. In this way, a strategy in two-phase improves the overall response time of the classification mechanism, facilitating a quick classification of those incoming SOAP messages with significative features during the first phase. The second phase is executed only for those SOAP messages with complex characteristics identified as suspicious during the first phase and requiring a more detailed evaluation. Each of the phases incorporates a CBR-BDI (Laza*, et al.*, 2003) agent with reasoning, learning and adaptation capabilities.

The approach presented in this study proposes a classifier agent for the first phase (CBRMAS-L1) that incorporates a classification strategy based on a classification tree, and a classifier agent for the second phase (CBRMAS-L2) that incorporates a neural network. Each of these classification strategies is incorporated into the respective re-use stage of the CBR cycle integrated into the corresponding agent. As a result, the system can learn and adapt to the attacks and the changes in the techniques used in the attacks. The model proposed in this study is innovative, since proposes a new perspective to address the XDoS attacks problem in Web services environments.

5.1.2.1 S-MAS Architecture

The architecture S-MAS presented in Figure 5.15 shows the four levels with BDI agents organized according to their roles.

- Traffic agents: Capture any traffic directed towards the server. Some Java classes of the TCPMon tool (Singh, 2010) were adapted to identify and capture any traffic that contains SOAP message packets. The captured SOAP messages are sent to the next layer in order to carry out the classification process. In addition to these tasks, Traffic agents use an IP register to monitor user activities. This type of monitoring makes it possible to identify suspicious activities similar to message replication attacks.

- CBRMAS-L1 agents: These advanced agents are the first-part of the core of the multi-agent architecture. These CBR-BDI agents are located on layer 2 of the architecture and are in charge of executing the first phase of the classification process based on the data sent by the Traffic agents. These agents initiate a classification by incorporating a CBR engine that in turn incorporates a decision tree strategy in the re-use phase. The main goal of this initial phase is to carry out an effective classification, but without requiring an excessive amount of resources.

- CBRMAS-L2 agents: These CBR-BDI agents complete the classification process from layer 3 of the architecture. These advanced agents are the

second-part of the core of the multi-agent architecture. In order to initiate this phase, it is necessary to have previously started a syntactic analysis on the SOAP message to extract the required data. This syntactic analysis is performed by the XMLAnalyzer Agent. Once the data have been extracted from the message, a CBR mechanism is initiated by using a Multilayer Perceptron (MLP) neural network in the re-use phase.

- Supervisor Agent: This agent supervises the XMLAnalyzer agent since there still exists the possibility of an attack during the syntactic processing of the SOAP message. This agent is located in layer 3 of the architecture.

- XMLAnalyzer Agent: This agent executes the syntactic analysis of the SOAP message. The analysis is performed using SAX (Brownell, 2002) as parser. Because SAX is an event driven API, it is most efficient primarily with regards to memory usage, and strong enough to deal with attack techniques. The data extracted from the syntactic analysis are sent to the CBRMAS-L2 agents. This agent is also located on layer 3 of the architecture.

- Coordinator Agent: This agent is in charge of supervising the correct overall functioning of the architecture. Additionally, it oversees the classification process. Each time a classification is tagged as suspicious, the agent interacts with the Interface Agent to request an expert review. Finally, this agent controls the alert mechanism and coordinates the actions required for responding to this type of attack. This agent is located on layer 4 of the architecture.

- Interface Agent: This agent was designed to function in different devices (PDA, Laptop, and Workstation). It facilitates ubiquitous communication with the security personnel when an alert has been sent by the Coordinator Agent. Additionally, it facilitates the process of adjusting the global configuration of the architecture. This agent is also located in the highest layer of the architecture.

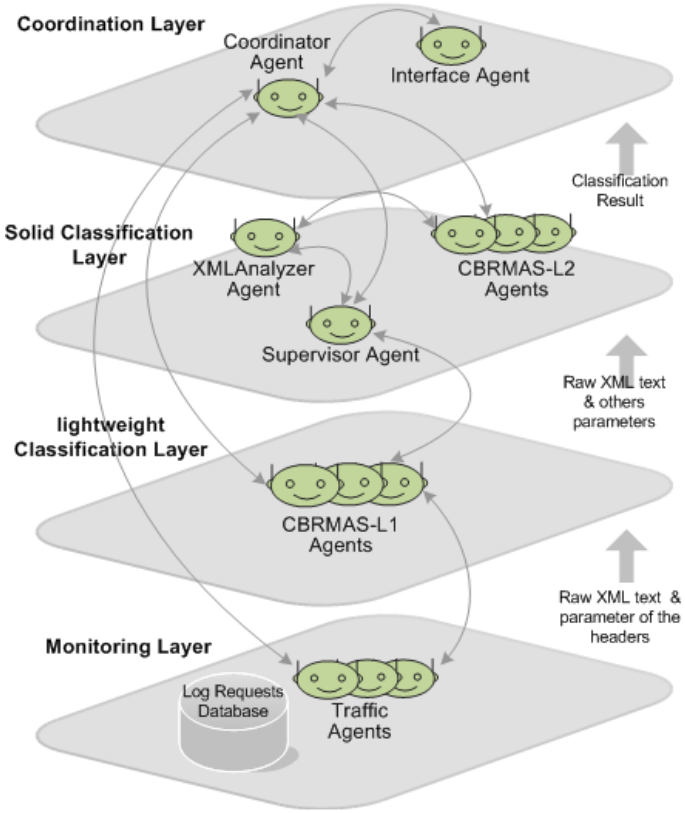

Figure 5.15. Design of the multi-agent architecture S-MAS proposed

The following section describes the functionality of the classifier agents located layers 2 and 3 of the proposed hierarchical structure.

5.1.2.2 Mechanism for the Classification of SOAP Message Attack

A CBR classifier agent (CBR-BDI) is responsible for classifying the incoming SOAP messages. A CBR engine requires the use of a database with which it can generate models such as the solution of a new problem based on past experience. In the specific case of SOAP messages, a case memory is managed for each service offered by the Web service environment, which permits it to handle each incoming message based on the particular characteristics of each Web service available. Each new SOAP message sent to the architecture is classified as a new

case study object. Focusing on the problem that is of interest to us, we will represent a typical SOAP message which consists of a type of wrapping that contains an optional heading and a mandatory body of text with a useful message load, as depicted in Figure 5.16-a and Figure 5.16-b.

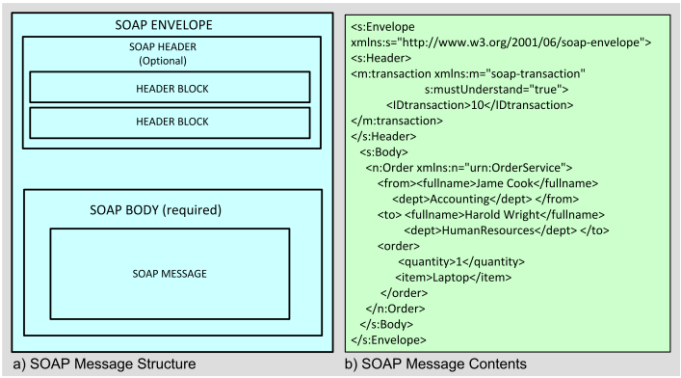

Figure 5.16. a) SOAP message structure (b) SOAP message content

Based on the structure and content of the SOAP message, the message processing tasks, and any activity among Web service users we can obtain a series of descriptive fields.

- First Phase of the mechanism of Classification – CBRMAS-L1 Agents

The main goal of this initial phase is to carry out an effective classification, but without requiring an excessive amount of resources and time. As a CBR strategy is used, it is necessary to define the case structure used by the CBRMAS-L1 agents. The fields of the case are obtained from the headers of the packages of the HTTP/TCP-IP transport protocol. Table 5.15 shows the fields taken into consideration to describe the problem.

Table 5.15. Problem description first phase – CBRSOAP-L1

Fields	Type	variable
IDService	Int	*i*
Subnet mask	String	*m*
SizeMessage	Int	*s*
NTimeRouting	Int	*n*

LengthSOAPAction	Int	*l*
TFMessageSent	Int	*w*

As can be seen in Table 5.15, the description of a case is given by the tuple $c = \left(i, m, s, n, l, w, R/c_{.im}, x^p, x^r\right)$, where *i* represents the service identifier, *m* the subnet mask, *s* the message length, *n* the number of seconds for the travel of the message, *l* the length of the header SoapAction, *w* the elapsed time from the arrival of the last *n* messages, $R/c_{.im}$ is the solution provided by the decision tree associated to the service and to the subnet mask, *xp* represents the class predicted by the CBR strategy $x^p \in X = \{a, g, u\}$, where *a*, *g*, *u* represent the values attack, good and undefined, *xr* is the real class $x^r \in X = \{a, g, u\}$.

The CBR strategy is integrated into a BDI agent, obtaining a CBR-BDI agent. The integration of the CBR system and the BDI agent is defined as follows: believes – problem description and rules; intentions – set of believes and rules that represent the state transitions required to achieve the final state; desires – $X = \{a, g, u\}$. The initial state is defined by means of the set of believes that store the values for the subnet mask and the Web service identifier, $(i, m, \phi, \phi, \phi, \phi)$. The intermediate states describe the decision process executed, taking into account the application of rules over the set of rules.

The cases memory contains a set of cases *C={c}* and is fragmented for each of the Web services available in the server. This structure facilitates the depuration and analysis of the services in an independent manner. Separately to the cases memory, the agent incorporates a rules memory, constructed as a set of inductive rules defined as $R = \{r_1, \ldots, r_l\}$ with $r_i = (l_1 \wedge \ldots \wedge l_m) \rightarrow x_j$ where $l_s = (d_{ts}, o_s, \Re) / d_{ts} \in \{i, m, s, n, l, w, x^p, x^r\}, o_s \in O$, with $O = \{=, \neq, >, <, \leq, \geq\}$, $x_j \in X$. The rules memory is also fragmented for each of the services and for each of the subnet mask, in a way that *R/C.im* represents the rules associated to those cases belonging to the service *i* and the subnet mask *m*. For notation considerations, to identify a property of a case, we use the case, a point and the property. For example, $C_{j.m}$ represents the property *m* (subnet mask) of the case *j*.

When the agent receives a request to classify a new case C_{n+1}, a new execution of a CBR cycle is carried out. The following paragraphs describe the stages of a CBR cycle executed by a CBRMAS-L1 agent in charge of a first phase classification.

- Retrieve: During this stage, those cases associated to the requested Web service and the corresponding rules memory are retrieved. The storage and

recovery of rules from the rules memory facilitates a notably reduction of the process time for the classification. The retrieve strategy is carried out as follows:

- If there is not tree associated to the service and the subnet mask, then it is necessary to recover the cases for the service and the subnet mask:

$$c_{.im} = f_s(C) = \{c_{j.im} \in C / c_{j.i} = c_{n+1 \cdot i}, c_{j \cdot m} = c_{n+1 \cdot m}\} \quad (5)$$

  Where $C_{j.i}$ represents the case *j* and *i* the service identifier.

- The rules memory associated with the set of cases $R/c_{.im}$ is retrieved.

- Reuse: Knowledge extraction is especially important when complex algorithms that use hard computing techniques and that generate models in an automatic way are used. Human experts are much confident when they know exactly why or at least how a solution to a problem has been calculated. CART is a nonparametric statistical method for extraction of knowledge in classifications. The extracted information is represented in a binary decision tree, which allows individuals to be classified from the root node. Keeping the kind of dependent variable in mind, CART can be separated into two types: classification tree, if the dependent variable is categorical; and regression tree in the case of a continuous dependent variable.

  The reuse stage is only executed if not decision tree $R/c_{.im}$ associated to the cases $c_{.im}$ is available, and in order to do so, the rules are generated using the CART algorithm. $R/c_{.im} = CART(c_{.im})$ where $R/c_{.im}$ is the rules memory associated to the service identifier and to the subnet mask. The CART algorithm has been modified in order to have an automatic discretization of the values to a set of categories. The modification includes a first step to normalize the variable into the interval [0,1] and then, the values are discretized into one of the following categories depending on the closest value {very low=0.1, low=0.3, medium=0.5, high=, 0.7 y very high=0.9}. This way, the generation of rules using the CART algorithm is more efficient than working with a greater level of categories. The discretization is only carried out for the variables s, n, l, w.

- Revise: Once the set of rules has been retriever, the classification for the case $C_{n+1.im}$ is obtained using the set of rules that previously classified the elements of the same type $c_{n+1.x^p} = R/c_{.im}(c_{n+1})$. If $r_i \in R/c_{.im}$ then, it is the rule that classifies C_{n+1}. The new case is classified as follows:

- If $m_i > \mu_1 \,||\, \#\{c_j \in C_{r_j} / c_{j \cdot x^p} = u\} > \mu_2$ then, it is necessary to execute the CBR of the second phase. Where $C_{r_i} \subseteq C$ is the set of cases classified for r_i and m_i represents the percentage of misclassified cases of C_{r_j} using the rule r_i . # represents the number of elements of the set. The general idea is to verify if the error rate of the rule exceeds a certain threshold, and then, verify that the number of cases belonging to the set of elements classified using the rule not exceeds a certain threshold defined as a function of the total number of elements in C_{r_j}

- Else if $\#\{c_j \in C_{r_j} / c_{j \cdot x^p} = g\} / \#C_{r_j} > \alpha_g$ then the case is classified as good and the revision finishes.

- Else if $\#\{c_j \in C_{r_j} / c_{j \cdot x^p} = u\} / \#C_{r_j} > \alpha_s$ the case is classified as suspicious and the second phase classification mechanism is executed.

- Else if $\#\{c_j \in C_{r_j} / c_{j \cdot x^p} = a\} / \#C_{r_j} > \alpha_a$ the case is classified as attack and the revision finishes.

- Retain: If the set of rules was generated because it didn't previously exist, then $R / c_{.im}$ is stored in the rules memory if the classification obtained was good. If the classification was erroneous and the misclassification was detected by an expert or if the CBRMAS-L2 of the second phase was invoked, then it is necessary to regenerate the decision tree: $R / c_{.im} = CART(c_{.im} \cup c_{n+1})$. Figure 5.17 shows the stages of the CBR cycle for the CBRMAS-L1 agent.

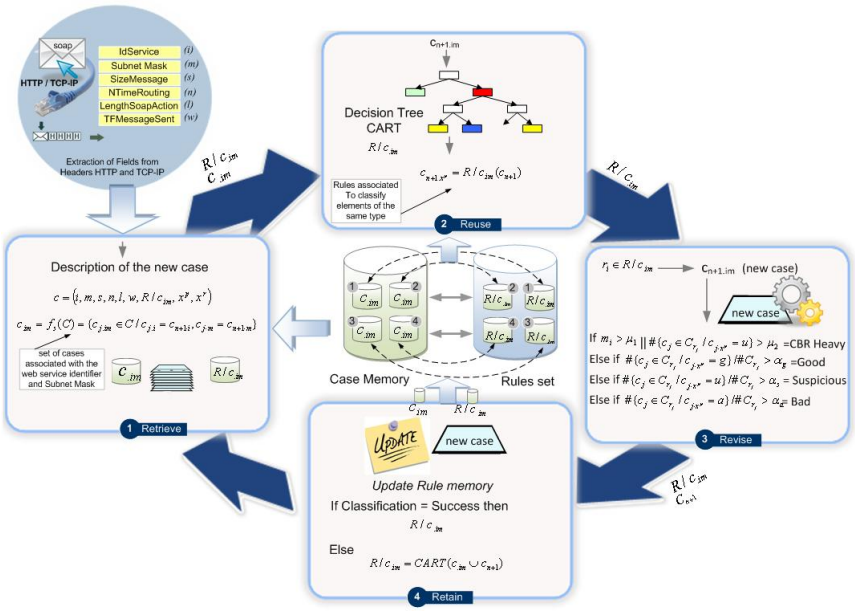

Figure 5.17. Stages of the CBR cycle of a CBRMAS-L1 agent in the first phase of the classification mechanism

- Second Phase of the mechanism of Classification – CBRMAS-L2 Agents

The second phase of the mechanism of classification is carried out by means of CBRMAS-L2 agents. As these agents are CBR-BDI agents, it is necessary to provide a case description. The fields are extracted from the SOAP message and provide the case description for the CBRMAS-L2 agents.

Table 5.16. Case description second phase – CBRSOAP-L2 agents

Fields	Type	variable
IDService	Int	*i*
MaskSubnet	String	*m*
SizeMessage	Int	*s*
NTimeRouting	Int	*n*
LengthSOAPAction	Int	*l*
MustUnderstandTrue	Boolean	*u*

NumberHeaderBlock	Int	*h*
NElementsBody	Int	*b*
NestingDepthElements	Int	*d*
NXMLTagRepeated	Int	*t*
NLeafNodesBody	Int	*f*
NAttributesDeclared	Int	*a*
CPUTimeParsing	Int	*c*
SizeKbMemoryParser	Int	*k*

Table 5.16 presents the fields used in describing the problem for the CBR in this layer. Applying the nomenclature shown in the table above, each case description is given by the following tuple:

$$c = \left(i, m, s, n, l, u, h, b, d, t, f, a, c, k, P/c_{\cdot im}, x^p, x^r\right) \quad (6)$$

For each incoming message received by the agent and requiring classification, we will consider both the class that the agent predicts and the class to which the message actually belongs. *xp* represents the class predicted by the CBRMAS-L2 agents belonging to the group. $x^p \in X = \{a, g, u\}$; *g* and *u* represent attack, good and undefined, respectively; and *xr* is the class to which the attack actually belongs $x^r \in X = \{a, g, u\}$, $P/c_{\cdot im}$ is the solution provided by the neural network MLP associated to service *i* and subnet mask *m*.

The reasoning memory used by the agent is defined by the following expression: $P = \{p_1, \ldots, p_n\}$ and is implemented by means of a MLP neural network. Each P_i is a reasoning memory related to a group of cases dependent of the service and subnet mask of the client. The Multilayer Perceptron (MLP) is the most widely applied and researched artificial neural network (ANN) model. MLP networks implement mappings from input space to output space and are normally applied to supervised learning tasks (Gallagher y Downs, 2003). The Sigmoidal function was selected as the MLP activation function, with a range of values in the interval [0, 1]. It is used to detect if the SOAP message is classified as an attack or not. The value 0 represents a legal message (non attack) and 1 a malicious message (attack). The sigmoidal activation function is given by

$$f(x) = \frac{1}{1 + e^{-ax}} \quad (7)$$

The CBR mechanism executes the following phases:

- Retrieve: Retrieves the cases that are most similar to the current problem, considering both the type of Web service to which the message belongs and the subnet mask that contains the message.
  - Expression 3 is used to select cases from the case memory based on the type of Web service and the subnet mask.

$$c_{.im} = f_s(C) = \{c_j \in C / c_{j.i} = c_{n+1 \cdot i}, c_{j \cdot m} = c_{n+1 \cdot m}\} \quad (8)$$

  - Once the similar cases have been recovered, the neural network MLP $P / c_{.im}$ associated to service *i* and subnet mask *m* is then recovered.

- Reuse: The classification of the message is begun in this phase, based on the subnet mask and the recovered cases. It is only necessary to retrain the neural network when it does not have previous training. The entries for the neural network correspond to the case elements *s*, *n*, *l*, *u*, *h*, *b*, *d*, *t*, *f*, *a*, *c*, *k*. Because the neurons exiting from the hidden layer of the neural network contain sigmoidal neurons with values between [0, 1], the incoming variables are redefined so that their range falls between [0.2-0.8]. This transformation is necessary because the network does not deal with values that fall outside of this range. The outgoing values are similarly limited to the range of [0.2, 0.8] with the value 0.2 corresponding to a non-attack and the value 0.8 corresponding to an attack. The training for the network is carried out by the error Backpropagation Algorithm (LeCun*, et al.*, 1998). The weights and biases for the neurons at the exit layer are updated by following equations:

$$w_{kj}^p(t+1) = w_{kj}^p(t) + \eta(d_k^p - y_k^p)(1 - y_k^p)y_k^p y_j^p + \mu(w_{kj}^p(t) - w_{kj}^p(t-1)) \quad (9)$$

$$\theta_k^p(t+1) = \theta_k^p(t) + \eta(d_k^p - y_k^p)(1 - y_k^p)y_k^p + \mu(\theta_k^p(t) - \theta_k^p(t-1) \quad (10)$$

The neurons at the intermediate layer are updated by following a procedure similar to the previous case using the following equations:

$$w_{ji}^p(t+1) = w_{ji}^p(t) + \eta(1 - y_j^p)y_j^p(\sum_{k=1}^{M}(d_k^p - y_k^p)(1 - y_k^p)y_k^p w_{kj})x_i^p + \mu(w_{ji}^p(t) - w_{ji}^p(t-1)) \quad (11)$$

$$\theta_j^p(t+1) = \theta_j^p(t) + \eta(1 - y_j^p)y_j^p(\sum_{k=1}^{M}(d_k^p - y_k^p)(1 - y_k^p)y_k^p w_{kj}) + \mu(\theta_j^p(t) - \theta_j^p(t-1)) \quad (12)$$

where w_{kj}^p represents the weight that joins neuron *j* from the intermediate layer with neuron *k* from the exit layer, *t* the moment of time and *p* the pattern in

question. d_k^p represents the desired value, y_k^p the value obtained for neuron *k* from the exit layer, y_j^p the value obtained for neuron *j* from the intermediate layer, η the learning rate and μ the momentum. θ_k^p represents the bia value *k* from the exit layer. The variables for the intermediate layer are defined analogously, keeping in mind that *i* represents the neuron from the entrance level, *j* is the neuron from the intermediate level, *M* is the number of neurons from the exit layer.

When a previously trained network is already available, the message classification process is carried out in the revise phase. If a previously trained network is not available, the training is carried out following the entire procedure beginning with the cases related to the service and subnet mask, as shown in equation (13).

$$p_r = MLP^t(c_{.im}) \tag{13}$$

- Revise: This phase reviews the classification performed in the previous phase. The value obtained by exiting the network $y = P_r^e(c_{n+1})$ may yield the following situations:
  - If $y > \mu_1$ then it is considered an attack
  - Otherwise, if $y < \mu_2$, then the message is considered a non-attack or legal
  - Otherwise, the message is marked as suspicious and is filtered for subsequent revision by a human expert. To facilitate the revision, an analysis of the neural network sensibility is shown so that the relevance of the entrances can be determined with respect to the predicted value.
- Retain: If the result of the classification is suspicious or if the administrator identifies the classification as erroneous, then the network $P/c_{.im}$ repeats the training by incorporating a new case and following the BackPropagation training algorithm.

$$p_r = MLP^t(c_{.im} \cup c_{n+1}) \tag{14}$$

Figure 5.18 shows the stages of the CBR cycle for the CBRMAS-L2 Agents, which constitute the second and last phase of the classification mechanism. The next section describes a case study developed to evaluate a prototype of the architecture presented in this study.

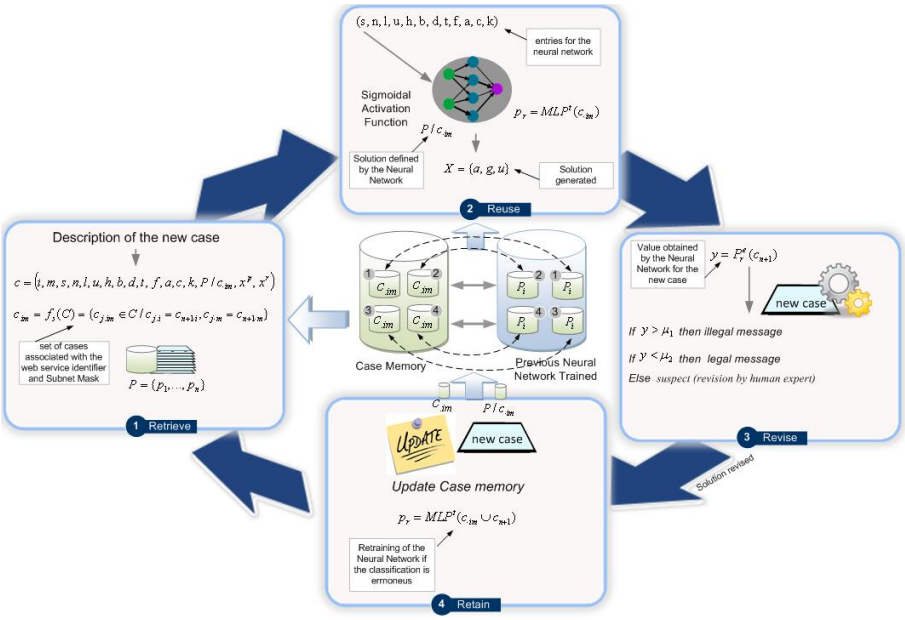

Figure 5.18. Stages of the CBR cycle of a CBRMAS-L2 agent in the first phase of the classification mechanism

5.1.2.3 Practical Application

A case study was proposed to test the effectiveness of a S-MAS prototype. An intelligent environment based on the use of Wi-Fi, Bluetooth and RFID and handheld devices was implemented in this mall. The intelligent environment improves the services offered in the shopping mall by providing personalized services through handheld devices. The clients can receive personalized promotions, recommendations about products or shops and guiding suggestions. They can also receive news or advises of their particular interest, or information about other clients with similar preferences (with whom they can communicate), as well as make use of indoor location services. The core of the intelligent environment is a CBP agent. The CBP agent attends to clients requesting suggestions. The clients then use their personal agents installed on their handheld devices (PDA, mobile phone, etc.) to interact with the intelligent environment. The CBP agent proposes guidance suggestions depending on client preferences and the shops´ capabilities.

The prototype implemented in this case study focused on the capacity to capture and classify SOAP messages in the shopping mall. To do this, the prototype incorporated the Traffic agents, the CBRMAS classifier agents and the XMLAnalyzer agent. These agents provide facilities to capture traffic, analyze the messages and provide a classification. The traffic agents were adapted to facilitate their download and installation in the handheld devices of the users in the shopping mall, in order to capture SOAP messages. The classifier agents were distributed between the PCs available for the experiment. Figure 5.19 presents the architecture of the system used to implement and evaluate the preliminary prototype for the approach presented in this study. As can be seen in Figure 5.19, the approach presented in this study to monitor and detect XML attacks is integrated within the previously existing multi-agent system in the shopping mall. Concretely, the security layer is located as an intermediate layer between the user's agents and the reasoning agents (planner and shop agents). In the inferior side of the image, it is possible to observe the different users that have access to the system using the Wi-Fi network of the mall or remote networks.

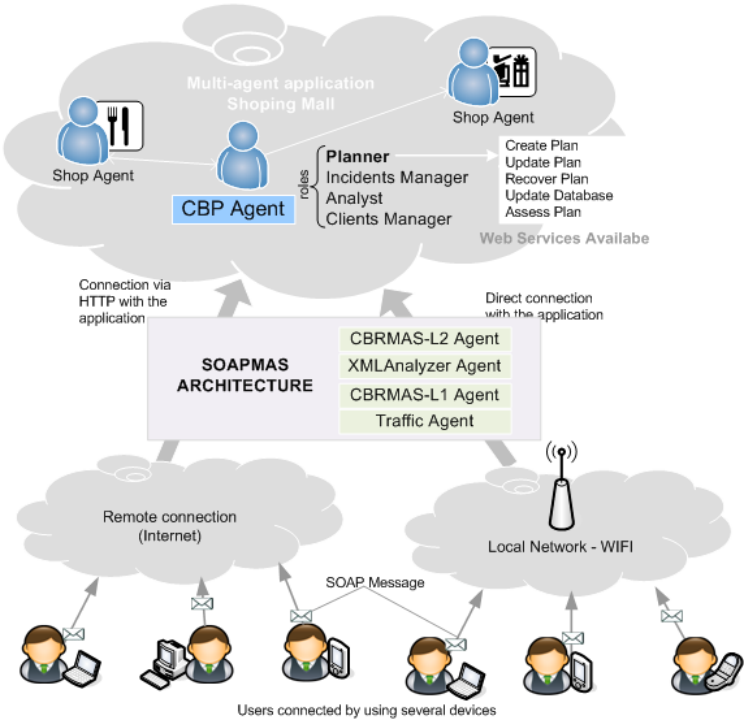

Figure 5.19. Scenario of the previous multi-agent system installed and the location of the S-MAS

To evaluate our proposal it was necessary to determine the Web services available in the previous existing multi-agent system (Corchado*, et al.*, 2009). As

the solution was based in a multi-agent architecture, it is possible to easily obtain the roles that play the agents as well as the Web services that will be offered in the system. This information is shown in Table 5.17.

Table 5.17. Agents with the roles and Web services available

CBP Agent with the roles and Web services available			
Planner	**Incidents Manager**	**Analyst**	**Clients Manager**
createPlan	addIncident	addUser	userList
modifyPlan	modifyIncident	analyzeSales	identifyUser
recoverPlan	deleteIncident	analyzePromotions	closeSesion
updateDatabase	queryIncident	addSurvey	
evaluatePlan	queryIncidents	analyzeSurvey	
		modifySurvey	
		querySurvey	
		querySurveys	
User Agent with the roles and Web services available			
Communicator	**Finder**	**Profile Manager**	
sendMessage	userLocation	addUser	
		deleteUser	
		modifyUser	
		queryUser	
		queryUsers	
Shop Agent with the roles and Web services available			
Promotions Manager	**Store Operator**		
addPromotion	addProduct		
deletePromotion	deleteProduct		
modifyPromotion	modifyStock		
queryPromotion	queryStock		
queryPromotions			

In the case study, we have focused on the CBP Planner agent, and specifically in the planner role, because it is the agent with a greater number of services available and that deals with a high number of variability in the messages, since the number of elements in the messages varies depending on the request type. Nowadays, the system is installed and working in the mall and, although the results obtained are satisfactory, certain technical problems related to the Bluetooth network were detected. An average of 65 clients daily connects to the system through their handheld devices to make use of the services provided in the mall. The average number of queries processed by the CBP agent varies between 12 and 16 per user, which generates an amount of 1040 to 1200

requests per day. The users' requests are mainly aimed at obtaining plans oriented to facilitate routes for shopping. The generation of routes requires to optimize the time available, to modify the plans in real-time and recover of previously stored plans. Another important group of requests are those oriented to close active sessions and to identify users. Finally, another significative group of requests (over 65 daily request) are those aimed at resolving incidents, complete surveys, etc.

All the previous mentioned services require an open platform, with communications capabilities to connect clients, shoppers and directorship in the mall. All the requests from the users and processed in the multi-agent system are codified using SOAP messages. These SOAP messages can be sent from different devices, as PDAs, Smart Phone, laptops, etc. connected to the local network of the Shopping mall or via internet. The structure of the SOAP messages, for the Web services offered by the CBP agent in the planner role, is defined through the information presented in Table 5.18, Table 5.19, Table 5.20, Table 5.21 and Table 5.22. This information represents the data associated to each of the plans.

Table 5.18. Problem description structure

ProblemDescription	Object Type
Case Id	Integer
Initial Location	Coordinate
Client Profile	ClientProfile
Client Preferences	ArrayList of ProductPreference
Restrictions	ArrayList of Restriction

Table 5.19. Product preference structure

ProductPreference	Object Type
MinimumPrice	Float
MaximumPrice	Float
StartTime	Date
FinishTime	Date
Product Type	Integer
Shop Type	Integer

Table 5.20. Planning restrictions

Restrictions	Object Type
TotalTime	Time
TotalMoney	Float

Table 5.21. Plan structure

Plan	Object Type
CaseId	Integer
route	Route
Quality	Float

Table 5.22. Route definition for a guidance suggestion

Route	Object Type
shop	Shop
ArrivalTime	Time
ServiceTime	Time
RetailProducts	ArrayList of Product
NextShop	Route

- A client profile contains information about a client's personal data (gender, economic level, postal code, number of children, and date of birth) and interests, and retail data (retail time and frequency, monthly profit - both business and product).
- The global restrictions are applied on the whole plan and not on each of the individual shops.
- The restrictions contain information about the time and money available, presented in Table 5.20.
- A route is a list representing the suggestion presented to the client, available Table 5.21
- The route consists of various stages, each of which contains information such as the shop visited by the client, arrival time, the time spent in the shop, the products consumed by the client, and the next destination, presented in Table 5.22.

The structure of the SOAP messages for the Web services of a CBP agent playing the planner role is shown in Table 5.23. Table 5.23 presents the fields required for the services as inputs. Furthermore, the fields of the outputs that the services provide are also shown.

Table 5.23. Summary of the structure of the SOAP messages, taking into account the parameters used as inputs for the Web services and the parameters used as outputs.

Web Service	Input-Web Service	Output-Web Service
createPlan	Table 5.18, Table 5.19	
updatePlan	Table 5.18, Table 5.19, Table 5.20	Table 5.21, Table 5.22
recoverPlan	Table 5.18, Table 5.19, Table 5.20, Table 5.21	Table 5.21, Table 5.22
updateDatabase	Table 5.20, Table 5.21, Table 5.22	
assessPlan	Table 5.21, Table 5.22	

An important information to take into account for each of the SOAP messages is the variability of the length of the messages. The variability in the number of elements and the level of nesting in the SOAP messages depends on the type of operation that executes the Web service. Generalizing for the Web services to assess, the approximate size of the SOAP messages sent by users ranges varies from 30Kb to 950Kb. Once defined the structure of the SOAP messages for a CBP agent playing the planner role, the next step in the evaluation of the solution is to set a timeframe for monitoring the execution environment and to provide information with input data to the prototype, and a period of 10 days can be considered as significative to evaluate the system, due to the significant volume of SOAP messages generated during a working day. As the prototype requires a memory of cases containing previous experiences, the first 5 days were used to obtain initial information and the last 5 days to test the effectiveness of the solution. The procedure was developed by capturing the incoming SOAP messages independently of their origin, the local network or remote access via the Internet. This would facilitate later generation of the memories of cases taking into account the subnet masks. During the first 5 days a total of 1231 SOAP messages were obtained (after filtering non-relevant messages). Table 5.24 details the distribution of the SOAP messages obtained for each of the Web services provided by the CBP agent.

Table 5.24. SOAP messages captured by each of the Web services of the planner role

Web Service	Total Messages
createPlan	185

updatePlan	326
recoverPlan	441
updateDatabase	123
assessPlan	156

The messages retrieved were SOAP messages under normal operating conditions (legal messages). However, our approach requires malicious messages to identify when a message corresponds to an attack or not. To achieve this goal, we developed a set of illegal SOAP messages on the basis of the knowledge about the structure of the Web services and SOAP messages used in the system. A set of 325 maliciously SOAP messages were generated, obtaining a total of 1556 SOAP messages. The distribution of the maliciously SOAP messages is presented in Table 5.25.

Table 5.25. Distribution of the total of malicious SOAP messages

Web Service	Malicious Messages
createPlan	110
updatePlan	35
recoverPlan	85
updateDatabase	73
assessPlan	22

With this initial information it was possible to generate the memories of cases for each of the CBRMAS agents used in the phases of the classification mechanism. Once the cases memories were generated, the evaluation was carried out in the following 5 days. At this stage, we consider a specific number of users. The users selected for the evaluation sent queries during 5 days that were captured and analyzed by the classification mechanism. The queries of the users came both from the local network and form the Internet. As the clients in the mall submit legal queries, we decided to introduce malicious SOAP messages in order to check the classification mechanism. These malicious SOAP messages were launched from the local network and from different remote Internet locations (different networks). During the 5 days of the tests, a total number of 1065 legal SOAP messages and 300 malicious SOAP messages were processed, which makes a total of 1365 messages.

Finally, to conclude the description of the case study some technical aspects of equipment used to conduct the tests are provided. These aspects are an influential factor in the results obtained, since the performance of the system is a critical factor when assessing this type of approach. The prototype, and more specifically the mechanism of classification was tested using two standard PC connected via a 100Mbps Ethernet network, using a physical switch which in turn was connected to the local network in the mall to capture the SOAP messages sent by users. Each PC was an HP Pavilion Intel Core 2 Duo E7200 with

4GB RAM. The tasks for the classification mechanism were distributed among the two PC. Next section presents the results and the conclusions obtained.

5.1.2.4 Results and Conclusion of the Case Study

SOAP messages used for communication in Web services environments are vulnerable to XDoS attacks. A XDoS attack launched on a Web services environment is a potential threat and can severely compromise the availability of the Web services. A new approach is presented in this article based on a new hierarchical multi-agent distributed architecture, S-MAS, for blocking malicious SOAP messages. S-MAS, unlike the centralized solutions (Im y Song, 2005) (Bebawy*, et al.*, 2005), (Wang, 2006), (Loh*, et al.*, 2006), (Gruschka y Luttenberger, 2006), (Padmanabhuni*, et al.*, 2006), (Yee*, et al.*, 2007), (Srivatsa*, et al.*, 2008), (Ye, 2008), (Chonka*, et al.*, 2009), is an adaptive approach that combines the advantages of multi-agent systems, such as autonomy and distributed problem solving (Corchado*, et al.*, 2008a), with the adaptation and learning capabilities of CBR systems (Corchado*, et al.*, 2003), (Corchado y Laza, 2003). Moreover, the user of decision trees and neural networks provides prediction and classification abilities, and their combination improves the overall functioning of the system.

The core of the architecture presented in this study is the two-phases classification mechanism specifically designed to analyze and classify the SOAP messages. The two-phases mechanism provides a hybrid alternative in which a first classifier filters the messages with a low-resource consumption and provides an initial classification. The second classifier is only executed when the first classifier couldn't provide a reliable result. The second classifier provides a high reliable classification, but with high-resource and time consumption and uses as input data the elements obtained from the SOAP messages after a syntactic analysis of the structure and XML content of the messages.

The solution presented in this study provides a new alternative for dealing with XDoS attacks in Web services environments, even when not overlooked penalizing response time required to execute the second phase of the classification mechanism. However, this penalty in response time is offset by the success of the solution because the procedure implemented in this latest phase of the classification mechanism allows a robust classification. Another important point to note in our solution is the ability to provide distributed and hierarchical capabilities. The distributed approach and the hierarchical design of the architecture helps to distribute tasks and to clarify how these tasks are assigned according to the roles of the different agents in each layer of the architecture, not forgetting the added capabilities offered by hierarchical structure for easy errors

recovery. When an agent in a concrete layer is compromised, the agent is eliminated and a new instance is created without affecting the rest of agents in the same layer or other layers of the architecture. Finally, a classification into two phases allows automatic solving of most of the incoming SOAP messages, reducing the possible intervention of a human expert to solve the final classification.

To check the validity of the proposed model, we elaborated a series of tests. The results obtained were promising, which allows us to conclude that S-MAS can be considered a good alternative for detecting and blocking of malicious SOAP message. The tests were conducted as follows: The first step was to generate the cases memories for the CBR engine of the CBRSOAP agents in every phase of the classification mechanism. Memories are made of cases involving the legal messages as well as maliciously messages. The cases memories were generated from the messages received in the first 5 days of monitoring the application installed in the mall, as explained in the case study. The next step was to test the prototype with a set of test performed with data obtained during the last 5 days of monitoring. The tests allow us to evaluate the global efficiency of the architecture by comparing different meaningful parameters before and after the implementation of the system in the test environment. The following paragraphs describe the experiments and discuss the conclusions obtained.

The first element to evaluate is the response time of S-MAS approach by analyzing the response time depending on the size of the messages. For this test we took a set of 200 messages of varying sizes and were entered into the system during the five testing days. Before entering the data into the system the information stored for those cases was deleted. As the system evolved during the days of testing, the response time is improved, but the message size directly affects the response time. The results obtained for this test are shown in Figure 5.20. In Figure 5.20, the x-axis shows the size of the SOAP messages (Kb), while the y-axis shows the response time in milliseconds (*ms*).

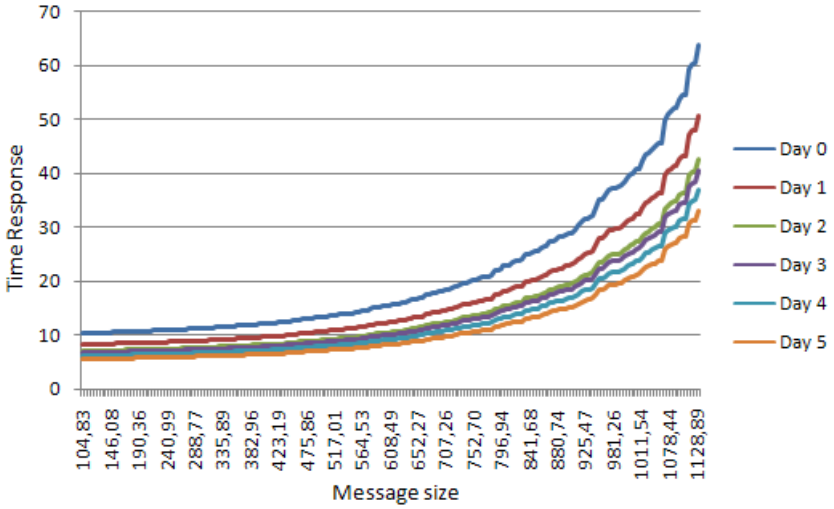

Figure 5.20. Evaluation of the response time depending on the size of the messages for the 5 testing days

The next element to evaluate is the percentage of false positives and false negatives generated by S-MAS, which are shown in Figure 5.21. The results obtained after the tests and shown in Figure 5.21 show that each of the radios represent the day of the test, and the breadth of the graph represents the percentage detected for the false positives and false negatives. In parentheses is represented the time lapse (in days) from the construction of the cases memories and the total number of cases stored.

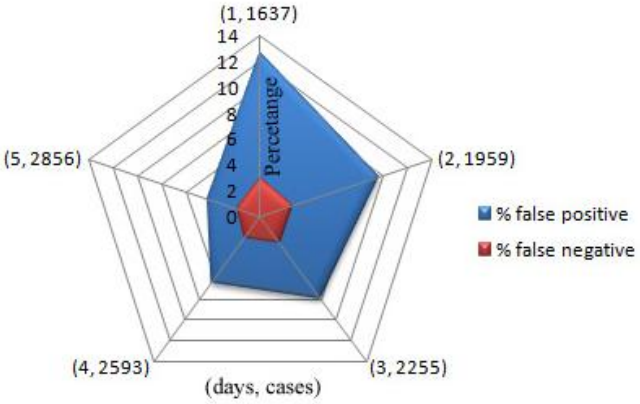

Figure 5.21. Percentage of false positives and false negatives detected in the system

To validate the evolution of the system, we proceeded to check the error rate as the cases were registered in the system. In Figure 5.22, it is possible to see the result obtained, and the decreasing trend in the system. The X axis represents the number of cases and the Y axis the percentage of errors found for the number of cases. It is clear that a large number the pattern of training improves the percentage of prediction. As we are working with CBR systems, which depend on a larger amount of data stored in the cases memory for each user, the percentage of success in the prediction increased. CBR systems need to draw from initial information (past experiences) in order to generalize efficient results.

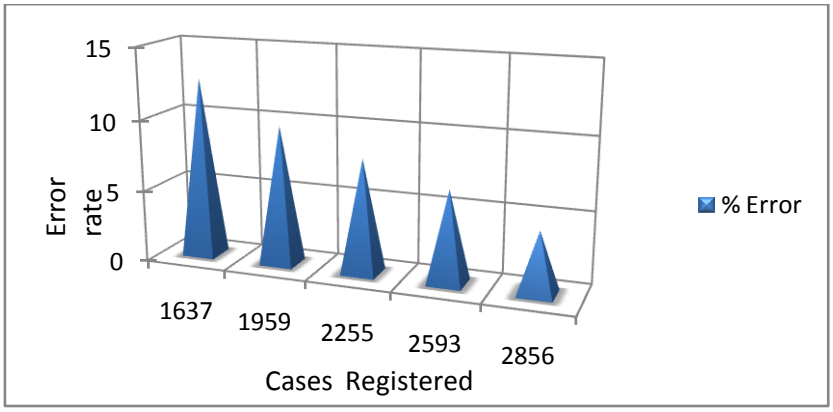

Figure 5.22. Error rate depending on the number of registered cases

As the number of cases in the cases memory of the CBRMAS-L1 agent increases, the number of times that it is necessary to execute a CBR cycle of the CBRMAS-L2 agent decreases, and this fact produces a reduction of the total execution time of the system. In Figure 5.23 can be seen the percentage of execution for each of the CBRMAS agents along the 5 days following to the initiation of the system, and the average time in milliseconds (*ms*) obtained for the execution of the classification of the services. The X-axis shows the initial and subsequent results obtained during the 5 testing days and the Y-axis represents the percentage of use of each of the CBRMAS agents, and the evolution of the average time. As can be seen, the average time decreased as the percentage of the use of the CBRMAS-L1 increased.

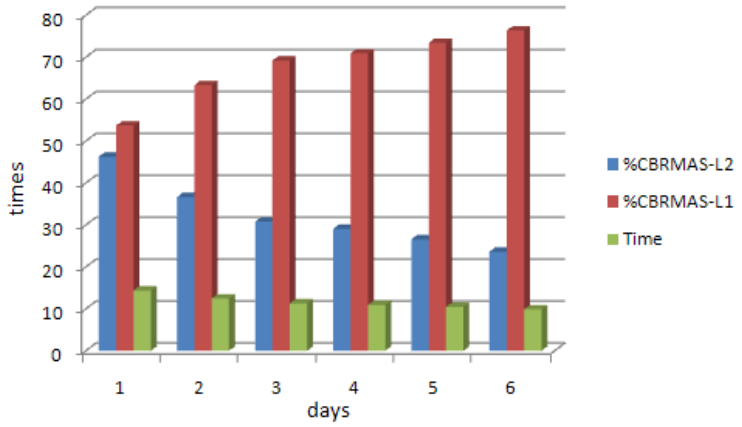

Figure 5.23. Percentage of execution for each of the CBRMAS agents along the 5 testing days and average execution time obtained for the classification of services

Finally, we examined the percentage of times that each of the CBRMAS agents were executed along the five days of testing. Figure 5.24 shows the results obtained for each of the CBRMAS agents, and, as can be seen, the percentage of times for both agents decreases significantly. The percentage of times that the decision tree is rebuilt is always higher than the neural network, since the neural network is only rebuilt when the second CBR is invoked and an error occurs, and the decision tree is always rebuilt when the second CBR is invoked. This significantly improves the performance of the system by decreasing the number of times required to train the neural network.

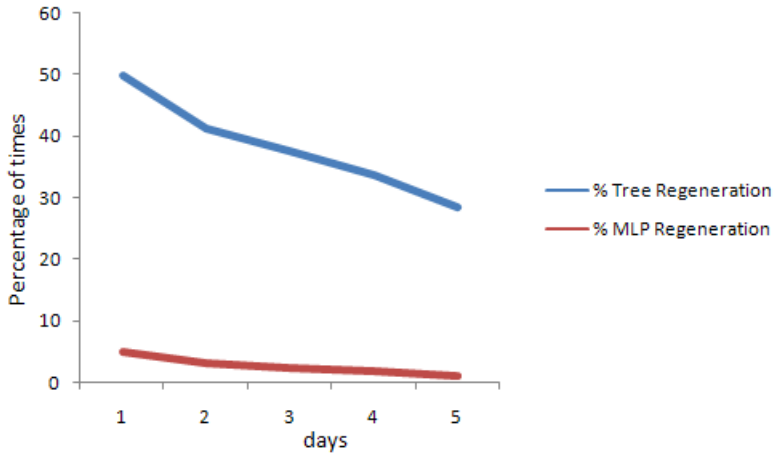

Figure 5.24. Percentage of times that each of the CBRMAS agents are executed

The architecture presented in this study provides a novel strategy for blocking malicious SOAP message. S-MAS evolved from the previous SCMAS architecture (Bajo*, et al.*, 2010), (Pinzón*, et al.*, 2008a) that was aimed at detecting and blocking SQL injection attacks. In this sense, S-MAS has improved the design of the SCMAS architecture facilitating a distributed perspective, as well as the distribution of services strategies. Moreover, S-MAS provides a novel classification method, since implements a two-phases classification, which notably improves the general performance of the system and reduces the response time with respect to SCMAS. Finally, the classification strategies have been adapted to the needs of the XML injection problem. As shown in this study, S-MAS proposes a new distributed perspective for detecting and preventing attacks and improves the functionalities offered by the existing approaches.

The results are promising and allow us to conclude that the S-MAS architecture considerably can be considered as a solid alternative to prevent and detect XDoS attacks in web service environments.

5.3 Conclusions

This chapter has presented two case studies where the AIDeMaS architecture has been applied to detect SQL injection and XDoS attacks. As shown, the AIDeMaS architecture has evolved over time and has been adapted to the detection of SQL injection and XDoS attacks. However, as demonstrated, the key component of the architecture, the CBR-BDI agents, remains invariable and provides advantages as learning and adaptation.

Both case studies have detailed the components of the architecture and their roles in the test scenarios. Moreover, the classification strategies have been analyzed and evaluated in detail, presenting the advantages of the detection strategies incorporated in the architecture.

The AIDeMaS architecture incorporates an innovative classification mechanism that facilitates the incorporation of different machine learning techniques and classification strategies. This mechanism provides advanced capacities to evaluate and select the more suitable techniques taking into consideration their effectiveness.

Finally, it is important to remark that the AIDeMaS architecture is presented as a solid and novel alternative in the field of denial of service attacks.

Capítulo 6

Conclusiones y Trabajo Futuro

Conclusión

En este trabajo de tesis doctoral se ha presentado una innovadora arquitectura multi-gente adaptativa para la detección de intrusiones en entornos distribuidos. Tal y como se ha expuesto a lo largo de esta memoria, en los últimos años, los sistemas distribuidos están transformando la forma de trabajar y de comunicarse. La seguridad se ha convertido en un área importante de investigación debido a las amenazas crecientes contra este tipo de sistemas. En este trabajo de investigación se han estudiado dos de los ataques más preocupantes en la actualidad: ataques de inyección SQL y los ataques XDoS. Estos ataques se caracterizan por una alta capacidad de evolucionar rápidamente, aplicando cambios en sus estrategias de ataque, y se centran principalmente en poner en riesgo la disponibilidad de los datos y las aplicaciones, bloqueando el acceso a los usuarios autorizados. La principal deficiencia observada en el estado del arte es que las medidas de seguridad existentes se centran en garantizar la confidencialidad e integridad de los datos, prestando poca atención a la disponibilidad.

Como respuesta a este problema, en este trabajo de tesis se ha presentado la arquitectura AIDeMaS, que se basa en la integración de la tecnología multi-agente, los sistemas de razonamiento basado en casos y técnicas del aprendizaje automático, para construir sistemas multi-agente para la detección de intrusiones en entornos distribuidos y dinámicos. Los sistemas multi-agente han resultado adecuados para resolver problemas en entornos dinámicos debido a sus capacidades inherentes, tales como su autonomía y sus capacidades de aprendizaje y razonamiento. Adicionalmente, los sistemas multi-agente facilitan el desarrollo de las tareas desde un enfoque distribuido. Por estas razones, la columna vertebral de la arquitectura AIDeMaS está basada en una arquitectura multi-agente diseñada a través de un modelo jerárquico en capas. La arquitectura define tipos de agentes especializados en la ejecución de tareas específicas y se encuentran situados en las distintas capas de la arquitectura en base a las tareas que se desarrollan en el proceso de detección de intrusiones. El componente clave de la arquitectura AIDeMaS es un mecanismo de clasificación novedoso sobre la base de agentes CBR-BDI. Los agentes CBR-BDI son agentes deliberativos que integran el paradigma del razonamiento basado en casos en su

estructura interna, mejorando así su capacidad de aprendizaje. Estos agentes dotan a la arquitectura de una gran capacidad de adaptación para hacer frente a los cambios de comportamiento que se producen en las estrategias de ataque.

Una de las innovaciones de la arquitectura AIDeMaS es que el proceso de detección de intrusiones se realiza aplicando técnicas de aprendizaje automático. La detección de intrusiones en la capa de aplicación de los sistemas distribuidos, resulta bastante complicada debido al enorme volumen de tráfico generado. Esta situación posibilita la penetración de intrusos a las aplicaciones camuflando los ataques con el tráfico legal. La cantidad de información que se requiere procesar, resulta muchas veces inmanejable para las técnicas tradicionales de detección de intrusiones. En este sentido, las técnicas y algoritmos del aprendizaje automático se presentan como una alternativa innovadora para resolver muchas de las limitaciones encontradas en las técnicas tradicionales. Concretamente, se hace uso de los arboles de decisión y las redes neuronales artificiales en las etapas del sistema de razonamiento basado en casos de los agentes CBR-BDI. En el caso del árbol de decisión implementado en el mecanismo de clasificación, a pesar de su simplicidad, permite filtrar aquellas peticiones maliciosas que siguen un patrón común de ataque sin un costo computacional alto. Por el contrario, la red MLP conlleva un coste computacional mayor pero provee una mayor efectividad con aquellas peticiones que no se ha podido resolver con el árbol de decisión debido a sus características más complejas. En resumen, ambas técnicas presentaron resultados prometedores como se pudo verificar en los casos de estudios presentados en el capítulo 5 de esta memoria de tesis doctoral.

De cara a remarcar las aportaciones de la arquitectura AIDeMaS con respecto al estado del arte se ha realizado una comparación con otras arquitecturas existentes centradas en la detección de ataques de inyección SQL y ataques XDoS, que fueron detalladas en el capítulo 2 de este trabajo de tesis doctoral. La comparación se realizó desde el punto de vista analítico, principalmente por la dificultad para tener acceso a las implementaciones de las arquitecturas existentes y, en algunos casos, las arquitecturas existentes solo están disponibles a nivel de propuesta. Para llevar a cabo la comparación, se realizó una revisión y evaluación minuciosa de los trabajos considerando que en algunos casos los artículos publicados, siendo el recurso más accesible, omiten detalles importantes por limitaciones de espacio, requiriendo entonces un análisis profundo para entender el funcionamiento y las generalidades del enfoque propuesto. Para evitar caer en errores en la comparación, en aquellos enfoques donde no fue posible obtener la información para confrontarlos con los aspectos a evaluar, los espacios correspondientes en la Tabla 6.1 fueron rellenados con un "-". Los aspectos a evaluar que pueden apreciarse en la Tabla 6.1 fueron extraídos principalmente de las características naturales de los sistemas de detección de intrusión. Para facilitar el manejo de la Tabla 6.1, los aspectos a evaluar (columna) han sido sustituidos con una variable, colocando entre paréntesis los valores disponibles. Los aspectos a evaluar comprenden: Tipo de ataques$^{(x1)}$ (Todos los mencionados en la Tabla 2.2 del capítulo 2 (todos, todos*,

algunos)), tipos de detección(x2) (basados en el uso indebido, anomalía, ambos), capacidad distribuida(x3) (Si, No), capacidad de aprendizaje(x4) (Si, No), capacidad de adaptación(x5) (Si, No), tolerancia a fallos(x6) (Si, No), escalabilidad(x7) (Si, No), precisión en la clasificación(x8) (alta, media, baja), Sobrecarga Computacional(x9) (Alta, Baja), ubiquidad(x10) (manejo de las alertas independiente del lugar(Sí, No)). Algunos detalles adicionales de la comparación son proporcionados para aclarar como los aspectos fueron considerados. En relación a los tipos de ataque, se consideró si el dato es proporcionado explícitamente dentro del trabajo; si este es el caso, el enfoque es evaluado rellenando el espacio en la tabla con la palabra "todos" o "algunos". Si no es proporcionado este dato, entonces se considera evaluando la estrategia de detección (técnicas o algoritmos usados e implementación); en ese caso el espacio en la tabla es rellenado con la palabra "todos*" acompañada de un asterisco para indicar que el resultado es obtenido evaluando la mecánica de detección, pero sin una demostración empírica. En el caso de la precisión en la clasificación nos referimos a la presencia o ausencia de falsos positivos y falsos negativos en el enfoque. Con respecto a la sobrecarga computacional tomamos en cuenta como se afecta el tiempo de respuesta. Para estos últimos aspectos, la información se tomó a partir de las evaluaciones de los casos de estudios y los resultados presentados en los trabajos.

Tabla 6.1. Comparación de los enfoques existentes con AIDeMaS en la detección de inyecciones SQL

Proposals	**X1**	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**	**X8**	**X9**	**X10**
(Bockermann, *et al.*, 2009)	Todos*	Anomalía	No	Sí	No	No	No	Media	Alta	No
(Ezumalai y Aghila, 2009)	Todos*	Uso Indebido	No	No	No	No	No	Alta	-	No
(Zhang, *et al.*, 2009)	Todos*	Uso Indebido	No	Sí	No	No	No	-	Baja	No
(Asmawi, 2008)	Todos*	Ambos	No	Sí	No	No	No	-	-	No
(Kemalis y Tzouramanis, 2008)	Todos	Uso Indebido	No	No	No	No	No	Alta	Baja	No
(Kiani, *et al.*, 2008)	Algunos	Anomalía	No	Sí	No	No	No	Media	Baja	No
(Bertino, *et al.*, 2007)	Todos*	Anomalía	No	Sí	No	No	No	Media	Baja	No
(Skaruz y Seredynski, 2007)	Todos*	Uso Indebido	No	Sí	No	No	No	Baja	-	No
(Robertson, *et al.*, 2006)	Todos*	Anomalía	No	Sí	No	No	No	Media	Baja	No
(García, *et al.*, 2006)	Todos*	Anomalía	No	Sí	No	No	No	Media	Baja	No
(Valeur, *et al.*, 2005)	Todos*	Anomalía	No	Sí	No	No	No	Baja	Baja	No
(Low, *et al.*, 2002)	Algunos	Uso Indebido	No	No	No	No	No	Media	Baja	No
AIDeMaS	Todos	Ambos	Sí	Sí	Sí	Sí	Sí	Media	Baja	Sí

Como se puede apreciar en la Tabla 6.1 los enfoques basados en IDS analizados, tienen como raíz la utilización de modelos basados en el propio aprendizaje automático. La solución que se propone en este trabajo, se fundamenta también en estrategias de Inteligencia Artificial que incorporan mecanismos para la representación del conocimiento y mecanismos de aprendizaje automático.

Además, la arquitectura ha sido comparada con las arquitecturas existentes para la detección de ataques de inyección XML. Aplicando la misma estrategia de comparación utilizada en la Tabla 6.1, se ha realizado una comparación de la arquitectura AIDeMaS con los enfoques existentes para hacer frente a los ataques XDoS. Los aspectos a evaluar fueron los siguientes: Capacidad distribuida(X1) (Si, No), capacidad de aprendizaje(X2) (Si, No), capacidad de adaptación(X3) (Si, No), tolerancia a fallos(X4) (Si, No), escalabilidad(X5) (Si, No), precisión en la clasificación(X6) (alta, media, baja), Sobrecarga Computacional(X7) (Alta, Baja), ubiquidad(X8) (manejo de las alertas independiente del lugar (Sí, No)).

Tabla 6.2. Comparación de los enfoques existentes y AIDeMaS en la detección de ataques XDoS

	X1	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**	**X8**
(Chonka, *et al.*, 2009)	Sí	Sí	No	-	Sí	Media	-	No
(Ye, 2008)	No	No	No	-	Sí	-	Baja	No
(Srivatsa, *et al.*, 2008)	No	No	NO	Sí	-	-	Baja	No
(Ye, 2008)	No	Sí	Sí	-	Sí	-	-	No
(Gruschka y Luttenberger, 2006)	No	No	No	Sí	No	-	Baja	No
(Loh, *et al.*, 2006)	No	No	No	-	Sí	Alta	-	No
(Bebawy, *et al.*, 2005)	No	No	No	-	Sí	-	-	No
(Im y Song, 2005)	No	No	No	No	No	-	-	No
AIDeMaS	Si	Si	Si	Si	Si	Media	Baja	Sí

Tal y como se puede apreciar en la Tabla 6.2 la mayoría de las propuestas revisadas se centran en modelos basados en cortafuegos, configurando y aplicando distintos tipos de políticas sobre los mensajes XML y ejecutando validaciones gramaticales para detectar mensajes maliciosos. Pocos han considerado evaluar otras técnicas que permitan abordar el problema de los ataques XDoS desde una perspectiva diferente. En la propuesta presentada en este trabajo de tesis, se planta una visión novedosa para explotar la capacidad de algoritmos y técnicas de Inteligencia Artificial, y así ganar flexibilidad y adaptabilidad a la hora de encarar los ataques XDoS.

En conclusión, a diferencia de los enfoques existentes para la detección de los ataques de inyección SQL y los ataques XDoS, AIDeMaS mejora al resto de los enfoques en función de:

- Tipo de Detección: AIDeMaS se diseña para explotar las ventajas de las técnicas más conocidas en detección de intrusos. Emplea como un primer

filtro una detección mediante firmas (*misuse detection*) y como componente principal de clasificación una detección basada en anomalía (*anomaly detection*).

- Enfoque Distribuido: AIDeMaS está basada en una arquitectura multi-agente que puede ejecutar tareas derivabas de la clasificación de una manera distribuida.

- Capacidad Adaptativa: AIDeMaS incluye tipos de agentes inteligentes diseñados para aprender y adaptarse a los cambios en los patrones de ataque, nuevo tipos de ataques y cambios en el comportamiento de los usuarios.

- Escalabilidad: AIDeMaS has sido diseñada para distribuir la carga de las tareas de clasificación a través de las capas de la arquitectura jerárquica. Adicional, dependiendo de la carga de trabajo, AIDeMaS es capaz de crecer instanciando nuevos agentes.

- Tolerancia a Fallos: AIDeMaS ha sido diseñada de forma jerárquica para limitar el impacto de errores y facilitar la recuperación de errores, eliminando de forma sencilla los agentes comprometidos.

- Ubicuidad: AIDeMaS provee un mecanismo de alerta ubicuo para notificar al personal de seguridad el evento de un ataque. Maneja un tipo de agente capaz de trabajar en dispositivos móviles.

Algunos aspectos de AIDeMaS, tales como el tiempo de respuesta o la curva de aprendizaje inicial puede ser una desventaja con respecto a las soluciones existentes. Sin embargo, AIDeMaS proporciona una manera más eficiente de clasificación una vez que el sistema ha adquirido experiencia, además de mejorar el tiempo de respuesta. A través de los resultados obtenidos en los experimentos y de las comparaciones realizadas, podemos concluir que la arquitectura AIDeMaS presenta características novedosas que no han sido consideradas en los enfoques previos y que aportan características importantes para resolver diferentes tipos de ataques XDoS y ataques de inyección SQL.

A continuación se presentan las contribuciones alcanzadas en esta investigación.

6.1 Contribuciones de la Investigación

Este trabajo aporta nuevas contribuciones en el ámbito de la detección de intrusiones, más concretamente en los entornos de las aplicaciones distribuidas.

Principalmente, la mayor novedad aportada consiste en una arquitectura multi-agente con un diseño jerárquico en capas, que incorpora en su estructura interna un mecanismo de clasificación adaptativo. A continuación se resumen las aportaciones realizadas en este trabajo de tesis doctoral:

- Marco para el análisis y diseño de una arquitectura multi-agente que permita construir sistemas multi-agente para la detección de intrusiones. Se realiza una revisión de los conceptos de detección de intrusiones, los modelos de detección de intrusiones aplicados en la actualidad, sus ventajas y desventajas. Se revisan los trabajos previos que utilizan sistemas multi-agente aplicados en la detección de intrusiones, y más concretamente aquellos enfocados en la detección de los ataques de inyección SQL y XDoS.

- Marco para el análisis y diseño de algoritmos de clasificación que facilitan aprendizaje automático aplicado a la detección de intrusiones. Se ha realizado un análisis de técnicas de aprendizaje automático aplicadas en la detección de intrusiones. Para cada una de las técnicas estudiadas se ha evaluado su tasa de efectividad y su carga computacional.

- Se ha propuesto una arquitectura multi-agente aplicada a la detección de intrusiones. La arquitectura multi-agente propone un modelo jerárquico que facilita un diseño novedoso con respecto a las estrategias tradicionales para la detección de intrusiones, ya que permite un diseño en capas para limitar el impacto de errores y facilita una rápida recuperación de errores. Además, permite un mayor nivel de escalabilidad, al poder adicionar nuevas instancias de agente en las capas correspondientes sin necesidad de realizar modificaciones estructurales. La arquitectura AIDeMaS permite realizar las tareas del proceso de detección de intrusión desde un enfoque distribuido. Los resultados obtenidos en los experimentos permiten demostrar que abordar el problema de la detección de ataques de inyección SQL e inyección XDoS de forma distribuida permite evitar los problemas tradicionales de los sistemas centralizados e incrementar la eficiencia en el proceso de detección.

- Como componente clave de la arquitectura AIDeMaS se ha diseñado un mecanismo de clasificación adaptativo para clasificar las peticiones de usuario y determinar su fiabilidad. El mecanismo de clasificación está basado en agentes CBR-BDI. Este tipo de agente BDI utiliza el razonamiento basado en casos como mecanismo deliberativo. De esta forma, el agente soluciona los problemas que se le plantean basándose en experiencias pasadas y en la creencia de que situaciones similares tendrán soluciones similares. Los sistemas CBR-BDI han demostrado ser muy eficientes en problemas similares que requieren un alto nivel de aprendizaje y adaptación a los cambios que ocurren en el entorno. Adicionalmente, los agentes CBR-BDI incorporan técnicas del aprendizaje automático para identificar las intrusiones en las peticiones maliciosas. El campo del aprendizaje automático se presenta como un área de investigación muy prometedora en

la detección de intrusiones. Desde el punto de vista global de la arquitectura presentada, este trabajo de tesis supone un gran avance con respecto al estado del arte. En los casos de estudio presentados en el capítulo 5 se ha evaluado la eficiencia del mecanismo de clasificación, y los resultados obtenidos han demostrado que mejora a los mecanismos utilizados en la actualidad para afrontar este tipo de problemas.

- Aplicación de la arquitectura a problemas reales. Para comprobar la validez de la arquitectura desarrollada, se han presentado dos casos de estudios en escenarios reales, permitiendo evaluar la efectividad del mecanismo de clasificación en la detección de los ataques estudiados.

Finalmente, es importante mencionar que el trabajo de investigación se ha completado realizando un estudio del estado del arte de la teoría de agentes, sistemas multi-agente, sistema de razonamiento basado en casos y técnicas del aprendizaje automático. Como parte fundamental de la investigación, se han estudiado los mecanismos deliberativos aplicables a los agentes, las herramientas para la construcción de sistemas multi-agente, y la búsqueda de escenarios reales para evaluar la propuesta.

6.2 Trabajo Futuro

En este punto del trabajo de investigación, se puede decir que se han alcanzado los objetivos inicialmente propuestos. No obstante, en el transcurso de la investigación fueron surgiendo nuevas líneas de interés que se han propuesto como trabajo futuro. A continuación se mencionan algunas de ellas:

- A nivel de la arquitectura multi-agente propuesta:
    - En la capa Monitorización de la arquitectura, se dispone de un tipo de agente para realizar tareas de auditoría, sin embargo, solamente se han desarrollado las funciones básicas para dicho agente. Se puede considerar la posibilidad de extender su funcionalidad de auditoría utilizando técnicas de minería de datos más complejas, como es el caso de técnicas de clasificación, agrupamiento y análisis de secuencia para incrementar la capacidad de detección de intrusiones.
    - En la capa Administración, se localiza un tipo de agente Interfaz, el cual facilita la interacción con la persona encargada de la seguridad en el sistema informático. Sería deseable evaluar la posibilidad de extender la funcionalidad de este agente incorporando nuevos mecanismos de visualización que puedan ayudar al experto en el proceso de decisión a confirmar la presencia de intrusiones.

- En las diferentes capas de la arquitectura se dispone de un tipo de agente Control para gestionar la comunicación y la asignación de tareas dentro de cada capa. Sería interesante considerar la posibilidad de extender su funcionalidad utilizando mecanismos más avanzados de planificación de tareas.

- Otras líneas más generales de trabajo futuro

  - Considerar la incorporación de nuevos mecanismos para ayudar a mejorar los tiempos y la calidad de las respuestas. En este sentido se ha evaluado la posibilidad de utilizar agentes de tiempo real implementando mecanismos de razonamiento con tiempo acotado. Adicionalmente, se propone estudiar nuevos modelos para el mecanismo de clasificación, por ejemplo aplicando el concepto de la mixtura de expertos.

  - En relación a la seguridad, la arquitectura implementa una capa de seguridad basada en la comunicación mediante el protocolo SSL (*secure socket layer*). Parece adecuado implementar nuevos mecanismos para garantizar las comunicaciones seguras y también el no repudio de las partes involucradas en la comunicación. En este caso se plantea investigar el campo de las firmas digitales.

  - Resolución de nuevos problemas en diferentes entornos reales. Para comprobar la validez de la arquitectura propuesta, se debe considerar su aplicación a nuevos casos de estudio. Esto permitiría comprobar si la arquitectura construida se adapta de forma adecuada a la resolución de problemas de características distintas, o bien si se ha construido una arquitectura muy restringida a los problemas concretos que se han estudiado durante la realización de este trabajo.

  - Aplicación a otros tipos de ataque. Este trabajo se ha centrado en el estudio de los ataques de inyección SQL y los ataques XDoS. Sin embargo, el diseño de la arquitectura fue pensado para facilitar la adaptación a otros tipos de ataques, por ejemplo inyecciones LDAP, las cuales siguen el mismo principio que las inyecciones SQL. Otros ataques que pueden considerarse son las inyecciones XPath, y los ataques XSS (*Cross-site scripting)*.

Es importante mencionar que algunas de las líneas de investigación, propuestas como trabajo futuro, han tenido una primera aproximación. En este sentido se ha estado en colaboración con otros grupos de investigación afines al tema. Es el caso de la utilización de mecanismos de visualización[1] y la

[1] Grupo de Inteligencia Computacional Aplicada (GICAP), Universidad de Burgo

implementación de agentes de tiempo real implementando mecanismos de razonamiento con tiempo acotado[2].

[2] Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia

Anexo A

Prototipo de la Arquitectura AIDeMaS

En este apartado se presentan algunas de las capturas más importantes del prototipo de la arquitectura AIDeMaS. Las capturas presentan la interfaz principal del sistema y las diferentes opciones de configuración. En la Figura A.1 se presenta la interfaz principal del prototipo.

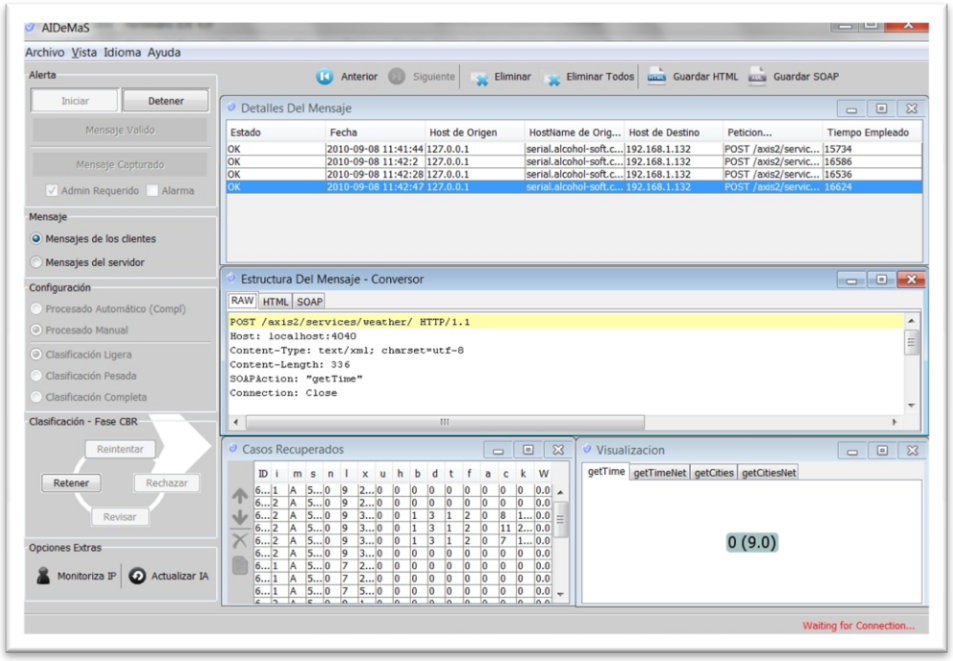

Figura A.1. Interfaz principal del prototipo

En la Figura A.2 se presenta una de las ventanas de la aplicación, donde se puede ver el contenido de la petición de usuario. En la Figura A.3 se presenta una visualización de la fase de clasificación pesada utilizando la red neuronal.

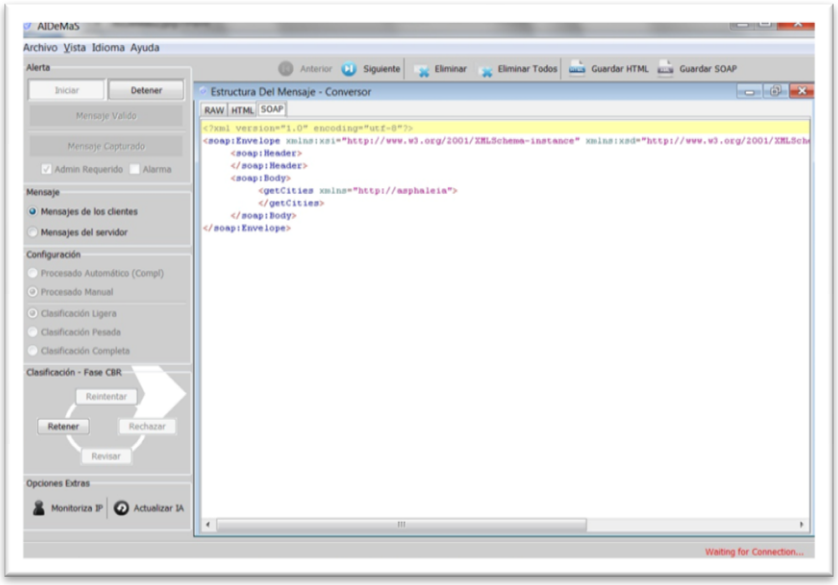

Figura A.2. Visualización del contenido de la petición de usuario

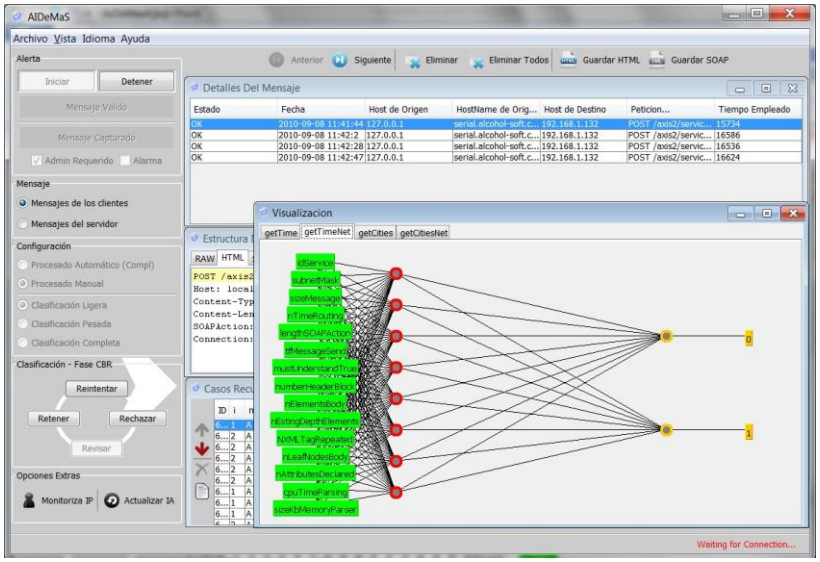

Figura A.3. Visualización de la clasificación pesada

En la Figura A.4 se presenta la interface para la monitorización de direcciones IP capturadas durante el envió de peticiones de usuario. La monitorización de direcciones IP nos permite filtrar aquellas direcciones IP que previamente han enviado peticiones maliciosas.

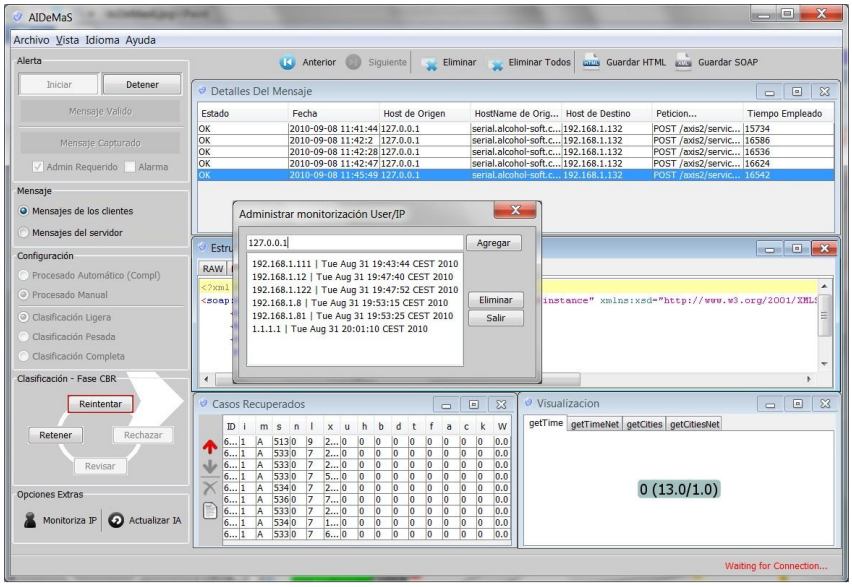

Figura A.4. Monitorización de direcciones IP.

Bibliografía

(Aamodt y Plaza, 1994) Aamodt, A., Plaza, E. (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications, 7*(1), 39-59.

(Abadeh*, et al.*, 2007) Abadeh, M. S., Habibi, J., Lucas, C. (2007). Intrusion detection using a fuzzy genetics-based learning algorithm. *Journal of Network and Computer Applications, 30*(1), 414-428. doi: j.jnca.2005.05.002

(Abraham*, et al.*, 2007) Abraham, A., Jain, R., Thomas, J., Han, S. Y. (2007). D-SCIDS: Distributed soft computing intrusion detection system. *Journal of Network and Computer Applications, 30*(1), 81-98. doi: 10.1016/j.jnca.2005.06.001

(Adaçal y Benner, 2006) Adaçal, M., Benner, A. B. (2006). Mobile Web Services: A New Agent-Based Framework. *IEEE Internet Computing, 10*(3), 58-65. doi: 10.1109/MIC.2006.59

(Aha*, et al.*, 1991) Aha, D. W., Kibler, D., Albert, M. K. (1991). Instance-Based Learning Algorithms. *Machine Learning, 6*(1), 37-66. doi: 10.1023/A:1022689900470

(Alpaydin, 2004) Alpaydin, E. (2004). *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*: The MIT Press.

(Allaire Security Bulletin, 1999) Allaire Security Bulletin. (1999). *Multiple SQL Statements in Dynamic Queries*. Retrieved from http://www.securityfocus.com/advisories/731

(Ambwani, 2003) Ambwani, T. (2003). Multi class support vector machine implementation to intrusion detection. In *International Joint Conference on Neural Networks* (Vol. 3, pp. 2300-2305). doi:10.1109/IJCNN.2003.1223795

(Anderson, 1980) Anderson, J. P. (1980). *Computer Security Threat Monitoring and Surveillance*. James P. Anderson Co. FortWashington, Pennsylvania,USA. Retrieved from http://csrc.nist.gov/publications/history/ande80.pdf

(Anderson*, et al.*, 2004a) Anderson, S., Bohren, J., Boubez, T., Chanliau, M., Della-Libera, G., Dixon, B. (2004a). *Web Services Secure Conversation Language (WS-SecureConversation) Version 1.1*. Retrieved from http://specs.xmlsoap.org/ws/2004/04/sc/ws-secureconversation.pdf

(Anderson*, et al.*, 2004b) Anderson, S., Bohren, J., Boubez, T., Chanliau, M., Della, G., Dixon, B. (2004b). *Web Services Trust Language (WS-Trust)*. Retrieved from http://www.ibm.com/developerworks/library/specification/ws-trust/

(Anley, 2002) Anley, C. (2002). *Advanced SQL Injection In SQL Server Applications*. Retrieved from http://www.nextgenss.com/papers/advanced_sql_injection.pdf

(Asaka, *et al.*, 1999) Asaka, M., Taguchi, A., Goto, S. (1999). The Implementation of IDA: An Intrusion Detection Agent System. In *11th FIRST Conference on Computer Security Incident Handling and Response*. Brisbane, Australia doi:10.1.1.38.7326

(Ashley y Brüninghaus, 2003) Ashley, K. D., Brüninghaus, S. (2003). A Predictive Role for Intermediate Legal Concepts. In *16th Annual Conference on Legal Knowledge and Information Systems, Jurix-03. 153. IOS* (pp. 153-162): Press.

(Asmawi, 2008) Asmawi, A., Sidek Zailani Mohamed Razak Shukor Abd. (2008). System architecture for SQL injection and insider misuse detection system for DBMS. In *International Symposium on Information Technology (ITSim'2008)* (Vol. 4, pp. 1-6). Kuala Lumpur, Malaysia doi:10.1109/ITSIM.2008.4631942

(Astley y Puppy, 1999) Astley, M., Puppy, R. F. (1999). *NT ODBC Remote Compromise*. Retrieved from http://www.securiteam.com/windowsntfocus/2AVQ5QAQKM.html

(Bajo y Corchado, 2005) Bajo, J., Corchado, J. M. (2005). Multiagent Architecture for Monitoring the North-Atlantic Carbon Dioxide Exchange RateJ. In Marín, R., Onaindía, E., Bugarín, A.,Santos, J. (Eds.), *Current Topics in Artificial Intelligence* (Vol. 4177, pp. 321-330): Springer Berlin / Heidelberg. doi:10.1007/11881216_34

(Bajo, *et al.*, 2010) Bajo, J., Corchado, J. M., Pinzón, C., De Paz, Y., Pérez-Lancho, B. (2010). SCMAS: A Distributed Hierarchical Multi-Agent Architecture for Blocking Attacks to Databases. *International Journal of Innovative Computing, Information and Control, 6*(9), 3787–3817.

(Bajo, *et al.*, 2006a) Bajo, J., de Luis, A., Gonzalez, A., Saavedra, A., Corchado, J. M. (2006a). A Shopping Mall Multiagent System: Ambient Intelligence in Practice. In Bravo, J. (Ed.), *2nd International Workshop on Ubiquitous Computing & Ambient Intelligence* (pp. 115-125)

(Bajo, *et al.*, 2006b) Bajo, J., De Luis, A., Tapia, D. I., Corchado, J. M. (2006b). Wireless Multi-Agent Systems based on CBR-BDI Agents: from Theory to Practice. In *5th International Workshop on Practical Applications of Agents and Multi-Agent Systems (IWPAAMS'06)* (Vol. 1, pp. 85-96)

(Bajo, *et al.*, 2007) Bajo, J., De Paz, J. F., Tapia, D. I., Corchado, J. M. (2007). Distributed Prediction of Carbon Dioxide Exchange Using CBR-BDI Agents. *International Journal of Computer Science*, 16-25.

(Bajo, *et al.*, 2006c) Bajo, J., De Paz, J. F., Tapia, D. I., Corchado, J. M. (2006c). Deliberative Agents for Distributed Monitoring and Evaluation of the Air-Sea Interaction. In Rezende, S. O.,da Silva Filho, A. C. R. (Eds.), *Industrial Applications of Distributed Intelligent Systems (INADIS'06)* (pp. 23-28)

(Bajo, *et al.*, 2008) Bajo, J., Julián, V., Corchado, J. M., Carrascosa, C., de Paz, Y., Botti, V., de Paz, J. F. (2008). An execution time planner for the ARTIS agent architecture. *Engineering Applications of Artificial Intelligence, 21*(5), 769-784. doi: 10.1016/j.engappai.2007.07.006

(Bandhakavi, *et al.*, 2007) Bandhakavi, S., Bisht, P., Madhusudan, P., Venkatakrishnan, V. N. (2007). CANDID: preventing sql injection attacks using dynamic candidate evaluations. In *14th ACM conference on Computer and communications security (CCS '07)* (pp. 12-24). New York, NY, USA: ACM. doi:10.1145/1315245.1315249

(Barbara, *et al.*, 2001) Barbara, D., Wu, N., Jajodia, S. (2001). Detecting Novel Network Intrusions using Bayes Estimators. In *First SIAM Conference on Data Mining*

(Bauer y Huget, 2003) Bauer, B., Huget, M. P. (2003). *FIPA modeling: agent class diagrams. Working Draft, foundation for Intelligent Physical Agents*. Retrieved from www.auml.org.

(BBN Technologies, 2009) BBN Technologies. (2009). *Cougaar: Cognitive Agent Architecture*. Retrieved from http://cougaar.org/

(Bebawy, *et al.*, 2005) Bebawy, R., Sabry, H., El-Kassas, S., Hanna, Y., Youssef, Y. (2005). Nedgty: Web Services Firewall. In *IEEE International Conference on Web Services (ICWS'05)* (Vol. 6, pp. 597-601). Orlando, Florida

(Belapurkar, *et al.*, 2009) Belapurkar, A., Chakrabarti, A., Ponnapalli, H., Varadarajan, N., Padmanabhuni, S., Sundarrajan, S. (2009). *Distributed Systems Security: Issues, Processes and Solutions*: Wiley Publishing.

(Bellifemine, *et al.*, 2001) Bellifemine, F., Poggi, A., Rimassa, G. (2001). JADE: a FIPA2000 compliant agent development environment. In *Fifth international conference on Autonomous agents (AGENTS '01)* (pp. 216-217). Montreal, Quebec, Canada: ACM. doi:10.1145/375735.376120

(Beqiri, 2009) Beqiri, E. (2009). Neural Networks for Intrusion Detection Systems. In Jahankhani, H., Hessami, A. G.,Hsu, F. (Eds.), *Global Security, Safety, and Sustainability: 5th International Conference (ICGS3'09)* (Vol. 45, pp. 156-165): Springer Berlin Heidelberg. doi:10.1007/978-3-642-04062-7_17

(Bergmann y Stahl, 1998) Bergmann, R., Stahl, A. (1998). Similarity Measures for Object-Oriented Case Representations. In *4th European Workshop on Advances in Case-Based Reasoning (EWCBR '98)* (pp. 25-36). London, UK: Springer-Verlag.

(Bertino, *et al.*, 2007) Bertino, E., Kamra, A., Early, J. (2007). Profiling Database Applications to Detect SQL Injection Attacks. In *Performance, Computing, and Communications Conference (IPCCC'2007)* (pp. 449-458). New Orleans, LA, USA doi:10.1109/PCCC.2007.358926

(Bertino, *et al.*, 2004) Bertino, E., Leggieri, T., Terzi, E. (2004). Securing DBMS: Characterizing and Detecting Query Floods. In Zhang, K. a. Z., Yuliang (Ed.), *7th International Conference Information Security (ISC'2004)* (pp. 195-206): Springer Berlin / Heidelberg. doi:10.1007/978-3-540-30144-8_17

(Bertino y Sandhu, 2005) Bertino, E., Sandhu, R. (2005). Database Security-Concepts, Approaches, and Challenges. In *IEEE Transactions on Dependable and Secure Computing* (Vol. 2, pp. 2-19). Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/TDSC.2005.9

(Bittencourt y Clarke, 2003) Bittencourt, H. R., Clarke, R. T. (2003). Use of classification and regression trees (CART) to classify remotely-sensed digital images. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS '03)* (Vol. 6, pp. 3751-3753)

(Bockermann, *et al.*, 2009) Bockermann, C., Apel, M., Meier, M. (2009). Learning SQL for Database Intrusion Detection Using Context-Sensitive Modelling (Extended Abstract). In *6th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA '09)* (pp. 196-205). Berlin, Heidelberg: Springer-Verlag.

(Botia, 2003) Botia, J. (2003). *Introducción a los agentes software - MAS*. Retrieved from http://ants.dif.um.es/staff/juanbot/ml/files/20022003/agentes.pdf

(Boyd y Keromytis, 2004) Boyd, S. W., Keromytis, A. D. (2004). SQLrand: Preventing SQL Injection Attacks. In *2nd Applied Cryptography and Network Security (ACNS'04)* (Vol. 3089, pp. 292-302)

(Bratman, 1987) Bratman, M. E. (1987). *Intention, Plans, and Practical Reason*. Cambridge, MA: Harvard University Press.

(Bratman, *et al.*, 1988) Bratman, M. E., Israel, D. J., Pollack, M. E. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence, 4* (3), 349-355. doi: 10.1111/j.1467-8640.1988.tb00284.x

(Bravenboer, *et al.*, 2007) Bravenboer, M., Dolstra, E., Visser, E. (2007). Preventing injection attacks with syntax embeddings. In *6th International conference on Generative programming and component engineering (GPCE '07)* (pp. 3-12). Salzburg, Austria: ACM. doi:10.1145/1289971.1289975

(Brenner, *et al.*, 1998) Brenner, W., Wittig, H., Zarnekow, R. (1998). *Intelligent Software Agents: Foundations and Applications*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

(Bridges, *et al.*, 2000) Bridges, S. M., Vaughn, R. B., Professor, A. (2000). Fuzzy Data Mining And Genetic Algorithms Applied To Intrusion Detection. In *National Information Systems Security Conference (NISSC)* (pp. 16-19). Baltimore, MD

(Brooks, 1991) Brooks, R. (1991). Intelligence Without Representation. *Artificial Intelligence, 47*, 139-159. doi: 10.1016/0004-3702(91)90053-M

(Brownell, 2002) Brownell, D. (2002). *SAX2*. Sebastopol, CA, USA: O'Reilly & Associates, Inc.

(Bruce y Dempsey, 1997) Bruce, G., Dempsey, R. (1997). *Security in distributed computing: did you lock the door?* Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

(Brugger, 2004) Brugger, S. T. (2004). *Data Mining Methods for Network Intrusion Detection.* University of California, Davis, California. Retrieved from http://www.bruggerink.com/~zow/GradSchool/brugger_dmnid.pdf

(Buehrer, *et al.*, 2005) Buehrer, G., Weide, B. W., Sivilotti, P. A. G. (2005). Using parse tree validation to prevent SQL injection attacks. In *5th international workshop on Software engineering and middleware (SEM*

'05) (pp. 106-113). New York, NY, USA: ACM. doi:10.1145/1108473.1108496

(Busquets*, et al.*, 2003) Busquets, D., Sierra, C., De Mántaras, R. L. (2003). A Multiagent Approach to Qualitative Landmark-Based Navigation. *Autonomous Robots, 15*(2), 129-154. doi: 10.1023/A:1025536924463

(Cansian*, et al.*, 1997) Cansian, A. M., Moreira, E., Carvalho, A., Bonifacio, J. M. (1997). Network intrusion detection using neural networks. In *International Conference on Computational Intelligence and Multimedia Applications (ICCMA'97)* (pp. 276–280). Gold Coast, Australia

(Carl*, et al.*, 2006) Carl, G., Kesidis, G., Brooks, R. R., Rai, S. (2006). Denial-of-Service Attack-Detection Techniques. *IEEE Internet Computing, 10*(1), 82-89. doi: 10.1109/MIC.2006.5

(Carrascosa*, et al.*, 2008) Carrascosa, C., Bajo, J., Julian, V., Corchado, J. M., Botti, V. (2008). Hybrid multi-agent architecture as a real-time problem-solving model. *Expert Systems with Applications, 34*(1), 2-17. doi: 10.1016/j.eswa.2006.08.031

(Cerami, 2002) Cerami, E. (2002). *Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL. First Edition* Sebastopol, CA, USA: O'Reilly & Associates.

(Cook y Rai, 2005) Cook, W. R., Rai, S. (2005). Safe query objects: statically typed objects as remotely executable queries. In *27th international conference on Software engineering (ICSE '05)* (pp. 97-106). St. Louis, MO, USA: ACM. doi:10.1145/1062455.1062488

(Corchado*, et al.*, 2007) Corchado, J. M., Aiken, I., Bajo, J. (2007). A CBP Agent for Monitoring the CO2 Exchange Rate. Case-Based Reasoning on Images and Signals. In Perner, P. (Ed.), *Studies on Computational Intelligence* (Vol. 73, pp. 213-246): Springer Verlag.

(Corchado*, et al.*, 2008a) Corchado, J. M., Bajo, J., Abraham, A. (2008a). GerAmi: Improving Healthcare Delivery in Geriatric Residences. *IEEE Intelligent Systems, 23*(2), 19-25. doi: 10.1109/MIS.2008.27

(Corchado*, et al.*, 2009) Corchado, J. M., Bajo, J., De Paz, J. F., Rodríguez, S. (2009). An Execution Time Neural-CBR Guidance Assistant. *Neurocomputing, 72*(13-15), 2743-2753. doi: 10.1016/j.neucom.2008.08.020

(Corchado*, et al.*, 2008b) Corchado, J. M., Glez-Bedia, M., De Paz, Y., Bajo, J., De Paz, J. F. (2008b). Replanning Mechanism For Deliberative Agents in Dynamic Changing Environments. *Computational Intelligence, 24*(2), 77-107. doi: 10.1111/j.1467-8640.2008.00323.x

(Corchado y Laza, 2003) Corchado, J. M., Laza, R. (2003). Constructing deliberative agents with case-based reasoning technology. *International Journal of Intelligent Systems, 18*, 1227-1241. doi: 10.1002/int.10138

(Corchado*, et al.*, 2003) Corchado, J. M., Laza, R., Borrajo, L., Yañez, J. C., De Luis, A., Valiño, M. (2003). Increasing the Autonomy of Deliberative Agents with a Case-Based Reasoning System. *International Journal of Computational Intelligence and Applications, 3*(1), 101-118. doi: 10.1142/S1469026803000823

(Corchado y Molina, 2002) Corchado, J. M., Molina, J. M. (2002). *Introducción a la Teoría de Agentes y Sistemas Multiagente*: Edite Publicaciones Científicas.

(Corchado*, et al.*, 2004) Corchado, J. M., Pavón, J., Corchado, E., Castillo, L. F. (2004). Development of CBR-BDI Agents: A Tourist Guide Application. In Funk, P., González, C.,Pedro, A. (Eds.), *Advances in Case-Based Reasoning* (Vol. 3155, pp. 51-54): Springer Berlin / Heidelberg. doi:10.1007/978-3-540-28631-8_40

(Cova*, et al.*, 2007) Cova, M., Balzarotti, D., Felmetsger, V., Vigna, G. (2007). Swaddler: An approach for the anomaly-based detection of state violations in web applications. In Kruegel, C., Lippmann, R.,Clark, A. (Eds.), *Recent Advances in Intrusion Detection* (Vol. 4637, pp. 63-86): Springer Berlin / Heidelberg.

(Coyle*, et al.*, 2004) Coyle, L., Doyle, D., Cunningham, P. (2004). Representing Similarity for CBR in XML. In *Advances in Case-Based Reasoning, 7th European Conference on Case Based Reasoning (ECCBR'04)* (pp. 119-127). Madrid, Spain: Springer Verlag.

(Chaib-draa y Dignum, 2002) Chaib-draa, B., Dignum, F. (2002). Trends In Agent Communication Language. *Computational Intelligence, 18*(2), 89-101. doi: 10.1111/1467-8640.00184

(Chakrabarti*, et al.*, 2008) Chakrabarti, A., Damodaran, A., Sengupta, S. (2008). Grid Computing Security: A Taxonomy. *IEEE Security and Privacy, 6*(1), 44-51. doi: 10.1109/MSP.2008.12

(Chandola*, et al.*, 2009) Chandola, V., Banerjee, A., Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys, 41*(3), 1-58. doi: 10.1145/1541880.1541882

(Chen*, et al.*, 2005) Chen, K., Chen, G., Dong, J. (2005). An Immunity-Based Intrusion Detection Solution for Database Systems. In Fan, W., Wu, Z.,Yang, J. (Eds.), *Advances in Web-Age Information Management, 6th International Conference Advances in Web-Age Information Management (WAIM'2005)* (Vol. 3739, pp. 773-778): Springer Berlin / Heidelberg. doi:10.1007/11563952_79

(Chen*, et al.*, 2009) Chen, Z., Wang, H., Abraham, A., Grosan, C., Yang, B., Chen, Y., Wang, L. (2009). Improving Neural Network Classification Using Further Division of Recognition Space. *International Journal of Innovative Computing, Information and Control, 5*(2), 301-310.

(Chonka*, et al.*, 2009) Chonka, A., Zhou, W., Xiang, Y. (2009). Defending Grid Web Services from XDoS Attacks by SOTA. In *EEE International Conference on Pervasive Computing and Communications* (Vol. 6, pp. 1-6). Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/PERCOM.2009.4912895

(Christensen*, et al.*, 2003) Christensen, A. S., Möller, A., Schwartzbach, M. I. (2003). Precise Analysis of String Expressions. In *10th International Static Analysis Symposium (SAS'03)* (Vol. 2694, pp. 1-18). San Diego, California, USA: Springer-Verlag.

(D. B. Networks, 2009) D. B. Networks. (2009). *SQL Injection Attack: Detection in a Web Application Environment*. Retrieved from

http://www.dbnetworks.com/pdf/sql-injection-detection-web-environment.pdf

(Dasgupta, *et al.*, 2005) Dasgupta, D., Gonzalez, F., Yallapu, K., Gomez, J., Yarramsettii, R. (2005). CIDS: An agent-based intrusion detection system. *Computers and Security, 24*(5), 387-398. doi: 10.1016/j.cose.2005.01.004

(De Paz, 2008) De Paz, Y. (2008). *Mixture of Weibull distributions by means of Artificial Neural Networks with censored data.* University of Salamanca.

(Debar y Dorizzi, 1992) Debar, H., Dorizzi, B. (1992). An application of a recurrent network to an intrusion detection system. *International Joint Conference on Neural Networks (IJCNN'02), 2*, 478-483. doi: 10.1109/IJCNN.1992.226942

(Del Brío y Molina, 2006) Del Brío, B. M., Molina, A. S. (2006). *Redes Neuronales y Sistemas Borrosos 3ra. Edición*. Madrid, España.

(Della-Libera, *et al.*, 2005) Della-Libera, G., Gudgin, M., Hallam-Baker, P., Hondo, M., Granqvist, H., Kaler, C., Maruyama, H., McIntosh, M., Nadalin, A., Nagaratnam, N., Philpott, R., Prafullchandra, H., Shewchuk, J., Walter, D., Zolfonoon, R. (2005). Web services security policy language Version 1.0 (WS-SecurityPolicy).

(Dickerson y Dickerson, 2000) Dickerson, J. E., Dickerson, J. A. (2000). Fuzzy network profiling for intrusion detection. In *19th International Conference of the North American:Fuzzy Information Processing Society (NAFIPS'00)* (pp. 301 -306)

(Dokas, *et al.*, 2002) Dokas, P., Ertoz, L., Kumar, V., Lazarevic, A., Srivastava, J., ning Tan, P. (2002). Data mining for network intrusion detection. In *National Science Foundation Workshop on Next Generation Data Mining* (pp. 21-30). Baltimore, MD

(Drogoul, *et al.*, 1995) Drogoul, A., Corbara, B., Lal, S. (1995). MANTA: New Experimental Results on the Emergence of (Artificial) Ant Societies. In *Artificial societies: The computer simulation of social life* (pp. 190-211). London: University College of London Press.

(Duda y Hart, 1973) Duda, R. O., Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. Storrs, CT, USA John Wiley & Sons Inc.

(Dunsmore y Brown, 2000) Dunsmore, B., Brown, J. (2000). *Mission Critical Internet Security (Mission Critical Series)*: Syngress Publishing.

(Ezumalai y Aghila, 2009) Ezumalai, R., Aghila, G. (2009). Combinatorial Approach for Preventing SQL Injection Attacks. In *IEEE International Advance Computing Conference (IACC'09)* (pp. 1212-1217). Patiala, India

(Fdez-Riverola, *et al.*, 2007) Fdez-Riverola, F., Iglesias, E. L., Díaz, F., Méndez, J. R., Corchado, J. M. (2007). SpamHunting: An instance-based reasoning system for spam labelling and filtering. *Decision Support Systems, 43*(3), 722-736. doi: 10.1016/j.dss.2006.11.012

(Ferguson, 1992) Ferguson, I. A. (1992). *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents.* University of Cambridge, Grange Road.

(Fernández, 2007) Fernández, G. E. (2007). *Wi-Fi: nuevos estándares en evolución*. Centro de Difusión de Tecnologías ETSIT-UPM. Retrieved from http://www.ceditec.etsit.upm.es/dmdocuments/wifi.pdf

(FIPA, 2007) FIPA. (2007). *FIPA: Foundation for Intelligent Physical Agents*. Retrieved from http://www.fipa.org/

(FIPA, 2002) FIPA. (2002). *FIPA-ACL Message Structure Specification*. Foundation for Intelligent Physical Agents. Retrieved from http://www.fipa.org/specs/fipa00061/SC00061G.pdf

(Foukia, 2005) Foukia, N. (2005). IDReAM: intrusion detection and response executed with agent mobility architecture and implementation. In *Fourth international joint conference on Autonomous agents and multiagent systemsn (AAMAS '05)* (pp. 264-270). New York, NY, USA: ACM. doi:10.1145/1082473.1082513

(Fox, *et al.*, 1990) Fox, K. L., Henning, R. R., Reed, J. H., Simonian, R. P. (1990). A neural network approach towards intrusion detection. In *13th National Computer Security Conference. Information Systems Security. Standards - the Key to the Future* (Vol. I, pp. 124-134). Gaithersburg, MD: NIST.

(Franklin y Graesser, 1997) Franklin, S., Graesser, A. (1997). Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents. In *Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages (ECAI '96)* (pp. 21-35). London, UK: Springer-Verlag.

(Fritzson, *et al.*, 1994) Fritzson, R., Finin, T., McKay, D., McEntire, R. (1994). KQML - A Language and Protocol for Knowledge and Information Exchange. In *13th International Distributed Artificial Intelligence Workshop*. Seattle WA.

(Fu, *et al.*, 2007) Fu, X., Lu, X., Peltsverger, B., Chen, S., Qian, K., Tao, L. (2007). A Static Analysis Framework For Detecting SQL Injection Vulnerabilities. In *31st Annual International Computer Software and Applications Conference (COMPSAC'07)* (pp. 87-96). Washington, DC, USA: IEEE Computer Society. doi:10.1109/COMPSAC.2007.43

(Fu y Qian, 2008) Fu, X., Qian, K. (2008). SAFELI: SQL injection scanner using symbolic execution. In *Workshop on Testing, analysis, and verification of web services and applications (TAV-WEB '08)* (pp. 34-39). New York, NY, USA: ACM. doi:10.1145/1390832.1390838

(Fujii, 2000) Fujii, K. (2000). Jpcap - a network packet capture library for applications written in Java. Retrieved from http://netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html

(Gallagher y Downs, 2003) Gallagher, M., Downs, T. (2003). Visualization of learning in multilayer perceptron networks using principal component analysis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 33*(1), 28-34. doi: 10.1109/TSMCB.2003.808183

(García, *et al.*, 2006) García, V. H., Monroy, R., Quintana, M. (2006). Web Attack Detection Using ID3. In *International Federation for Information Processing* (pp. 323-332). Santiago, Chile. doi:10.1007/978-0-387-34749-3_34

(Georgeff, *et al.*, 1998) Georgeff, M., Pollack, M., Tambe, M. (1998). The Belief-Desire-Intention Model of Agency. In *Fifth International Workshop on Agent Theories, Architectures and Languages (ATAL'98)*. London, UK: Springer-Verlag, Heidelberg.

(Georgeff y Lansky, 1987) Georgeff, M. P., Lansky, A. L. (1987). Reactive reasoning and planning. In *Sixth National Conference on Artificial Intelligence (AAAI'87)* (pp. 677-682). Seattle, Washington: AAAI Press.

(Golding y Rosenbloom, 1988) Golding, A. R., Rosenbloom, P. S. (1988). Combining Analytical and Similarity-Based CBR. In *2nd Case-Based Reasoning Workshop* (pp. 259-263). Pensacola Beach, Florida: Morgan Kaufman.

(Gould, *et al.*, 2004) Gould, C., Su, Z., Devanbu, P. (2004). JDBC Checker: A Static Analysis Tool for SQL/JDBC Applications. In *26th International Conference on Software Engineering (ICSE '04)* (pp. 697-698). Washington, DC, USA: IEEE Computer Society.

(Gowadia, *et al.*, 2005) Gowadia, V., Farkas, C., Valtorta, M. (2005). Paid: A probabilistic agent-based intrusion detection system. *Computers & Security, 24*(7), 529-545. doi: 10.1016/j.cose.2005.06.008

(Gruschka y Luttenberger, 2006) Gruschka, N., Luttenberger, N. (2006). Protecting Web Services from DoS Attacks by SOAP Message Validation. In Fischer-Hübner, S., Rannenberg, K., Yngström, L.,Lindskog, S. (Eds.), *Security and Privacy in Dynamic Environments* (Vol. 201, pp. 171-182): Springer Boston. doi:10.1007/0-387-33406-8

(Guan, *et al.*, 2004) Guan, J., xin Liu, D., Wang, T. (2004). Applications of Fuzzy Data Mining Methods for Intrusion Detection Systems. In Laganà, A., Gavrilova, M. L., Kumar, V., Mun, Y., Tan, C. J. K.,Gervasi, O. (Eds.), *Computational Science and Its Applications (ICCSA'04)* (Vol. 3045, pp. 706-714). Assisi, Italy. doi:10.1007/978-3-540-24767-8_74

(Gudgin, *et al.*, 2007) Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J. J., Nielsen, H. F., Karmarkar, A., Lafon, Y. (2007). *W3C Recommendation: SOAP Version 1.2 Part 2: Adjuncts (Second Edition)*. Retrieved from http://www.w3.org/TR/soap12-part2/

(Haldar, *et al.*, 2005) Haldar, V., Chandra, D., Franz, M. (2005). Dynamic Taint Propagation for Java. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC '05)* (pp. 303-311). Washington, DC, USA: IEEE Computer Society. doi:10.1109/CSAC.2005.21

(Halfond y Orso, 2005a) Halfond, W. G. J., Orso, A. (2005a). AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks. In *20th IEEE/ACM international Conference on Automated software engineering (ASE '05)* (pp. 174-183). New York, NY, USA: ACM. doi:10.1145/1101908.1101935

(Halfond y Orso, 2005b) Halfond, W. G. J., Orso, A. (2005b). Combining static analysis and runtime monitoring to counter SQL-injection attacks. In *Third international workshop on Dynamic analysis (WODA '05)* (pp. 1-7). St. Louis, Missouri: ACM. doi:10.1145/1083246.1083250

(Halfond, *et al.*, 2006) Halfond, W. G. J., Viegas, J., Orso, A. (2006). A Classification of SQL-Injection Attacks and Countermeasures. In *IEEE International Symposium on Secure Software Engineering*. Arlington, VA, USA

(Han, *et al.*, 2002) Han, H., Lu, X.-L., Ren, L.-Y. (2002). Using data mining to discover signatures in network-based intrusion detection. In *International Conference on Machine Learning and Cybernetics* (Vol. 1, pp. 13-17). doi:10.1109/ICMLC.2002.1176698

(Hayat, *et al.*, 2007) Hayat, Z., Reeve, J., Boutle, C. (2007). Ubiquitous security for ubiquitous computing. *Information Security Tech. Report, 12*, 172-178. doi: 10.1016/j.istr.2007.05.002

(Heaton, 2008) Heaton, J. (2008). *Introduction to Neural Networks for Java, 2nd Edition*: Heaton Research, Inc.

(Heberlein, *et al.*, 1990) Heberlein, L. T., Dias, G. V., Levitt, K. N., Mukherjee, B., Wood, J., Wolber, D. (1990). A Network Security Monitor. In *IEEE Symposium on Security and Privacy* (Vol. 0, pp. 296): IEEE Computer Society. doi:10.1109/RISP.1990.63859

(Hegazy, *et al.*, 2003) Hegazy, I. M., Al-Arif, T., Fayed, Z. T., Faheem, H. M. (2003). A Multi-agent Based System for Intrusion Detection. In *IEEE Potentials* (Vol. 22, pp. 28–31). doi:10.1109/MP.2003.1238690

(Helali, 2010) Helali, R. G. M. (2010). Data mining based network intrusion detection system: A survey. In Sobh, T., Elleithy, K.,Mahmood, A. (Eds.), *Novel Algorithms and Techniques in Telecommunications and Networking* (pp. 501-505). Dordrecht: Springer Netherlands. doi:10.1007/978-90-481-3662-9_86

(Helmer, *et al.*, 2002) Helmer, G., Wong, J. S. K., Honavar, V. G., Miller, L. (2002). Automated discovery of concise predictive rules for intrusion detection. *Journal of Systems and Software, 60*, 165-175. doi: 10.1016/S0164-1212(01)00088-7

(Helmer, *et al.*, 1999) Helmer, G. G., Wong, J. S., Honavar, V., Miller, L. (1999). Feature Selection Using a Genetic Algorithm for Intrusion Detection. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M.,Smith, R. E. (Eds.), *Genetic and Evolutionary Computation Conference (GECCO'99)* (pp. 1781). Orlando, FL, USA

(Hendler, 2006) Hendler, J. (2006). Introduction to the Special Issue: AI, Agents, and the Web. *IEEE Intelligent Systems, 21*(1), 11. doi: 10.1109/MIS.2006.11

(Hernandez, 2007) Hernandez, D. R. (2007). Introduction to the Bayesian Analysis. Mar del Plata: Publicaciones Especiales INIDEP.

(Herrero y Corchado, 2009) Herrero, Á., Corchado, E. (2009). Multiagent Systems for Network Intrusion Detection: A Review. In Herrero, Á., Gastaldo, P., Zunino, R.,Corchado, E. (Eds.), *Computational Intel. in Security for Info. Systems (AISC)* (Vol. 63, pp. 143–154): Springer-Verlag Berlin Heidelberg.

(Herrero, *et al.*, 2009) Herrero, Á., Corchado, E., Pellicer, M. A., Abraham, A. (2009). MOVIH-IDS: A mobile-visualization hybrid intrusion detection

system. *Neurocomputing, 72*(13-15), 2775-2784. doi: j.neucom.2008.12.033

(Holland, 1975) Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press.

(Huang, *et al.*, 2003) Huang, Y.-W., Huang, S.-K., Lin, T.-P., Tsai, C.-H. (2003). Web application security assessment by fault injection and behavior monitoring. In *12th international conference on World Wide Web (WWW '03)* (pp. 148-159). Budapest, Hungary: ACM. doi:10.1145/775152.775174

(Huang, *et al.*, 2004) Huang, Y.-W., Yu, F., Hang, C., Tsai, C.-H., Lee, D.-T., Kuo, S.-Y. (2004). Securing web application code by static analysis and runtime protection. In *13th international conference on World Wide Web (WWW '04)* (pp. 40-52). New York, NY, USA: ACM. doi:10.1145/988672.988679

(IBM Corporation, 2009) IBM Corporation. (2009). *IBM Internet Security Systems X-Force® 2008 Trend & Risk Report*. Retrieved from https://www-935.ibm.com/services/us/iss/xforce/trendreports/xforce-2008-annual-report.pdf

(Im y Song, 2005) Im, E. G., Song, Y. H. (2005). An Adaptive Approach to Handle DoS Attack for Web Services. In Kantor, P., Muresan, G., Roberts, F., Zeng, D. D., Wang, F.-Y., Chen, H.,Merkle, R. C. (Eds.), *Intelligence and Security Informatics* (Vol. 3495/2005, pp. 634-635): Springer Berlin / Heidelberg. doi:10.1007/11427995_83

(Jackson, *et al.*, 1990) Jackson, K. A., Dubois, D. H., Stallings, C. A. (1990). *NADIR - A Prototype Network Intrusion Detection System*. Los Alamos National Laboratory. Retrieved from http://www.osti.gov/bridge/servlets/purl/6192985-Ud770j/6192985.pdf

(JADE Board, 2005) JADE Board. (2005). *JADE Security Guide*. Retrieved from http://jade.cselt.it/doc/tutorials/JADE_Security.pdf

(Jazayeri, 2007) Jazayeri, M. (2007). Some Trends in Web Application Development. In *Future of Software Engineering (FOSE'07)* (pp. 199-213). Washington, DC, USA: IEEE Computer Society. doi:10.1109/FOSE.2007.26

(Jennings y Bussmann, 2003) Jennings, N. R., Bussmann, S. (2003). Agent-based control systems. *IEEE Control Systems, 23*(3), 61-74.

(Jennings, *et al.*, 1998) Jennings, N. R., Sycara, K., Wooldridge, M. (1998). A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems, 1*(1), 7-38. doi: 10.1023/A:1010090405266

(Jones, 2008) Jones, M. T. (2008). *Artificial Intelligence: A Systems Approach* Jones and Bartlett Publishers, Inc.

(Junjin, 2009) Junjin, M. (2009). An Approach for SQL Injection Vulnerability Detection. In *Sixth International Conference on Information Technology: New Generations (ITNG '09)* (pp. 1411-1414). Washington, DC, USA: IEEE Computer Society. doi:10.1109/ITNG.2009.34

(Kahn, *et al.*, 1998) Kahn, C., Porras, P. A., Staniford-chen, S., Tung, B. (1998). A common intrusion detection framework. *Journal of Computer Security*.

(Kandeeban y Rajesh, 2010) Kandeeban, S. S., Rajesh, R. S. (2010). Integrated Intrusion Detection System Using Soft Computing. *International Journal of Network Security, 10*, 87-92.

(Kemalis y Tzouramanis, 2008) Kemalis, K., Tzouramanis, T. (2008). SQL-IDS: a specification-based approach for SQL-injection detection. In *ACM symposium on Applied computing (SAC'2008)* (pp. 2153-2158). Fortaleza, Ceara, Brazil: ACM. doi:10.1145/1363686.1364201

(Kiani, *et al.*, 2008) Kiani, M., Clark, A., Mohay, G. (2008). Evaluation of Anomaly Based Character Distribution Models in the Detection of SQL Injection Attacks. In *Third International Conference on Availability, Reliability and Security (ARES'2008)* (pp. 47-55). Washington, DC, USA: IEEE Computer Society. doi:10.1109/ARES.2008.123

(Kinny y Georgeff, 1991) Kinny, D., Georgeff, M. (1991). Commitment and effectiveness of situated agents. In *Thirteenth International Joint Conference on Artificial Intelligence (IJCAI'91)* (pp. 82-88). Sydney, New South Wales, Australia: Morgan Kaufmann Publishers Inc.

(Klir y Yuan, 1995) Klir, G. J., Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Applications* (1st ed.). Upper Saddle River, NJ, USA: Prentice Hall PTR.

(Ko, *et al.*, 1997) Ko, C., Ruschitzka, M., Levitt, K. (1997). Execution monitoring of security-critical programs in distributed systems: a Specification-based approach. In *IEEE Symposium on Security and Privacy (SP '97)* (pp. 175-187). Washington, DC, USA: IEEE Computer Society. doi:10.1109/SECPRI.1997.601332

(Kolodner, 1993) Kolodner, J. (1993). *Case-Based Reasoning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

(Konheim, 2007) Konheim, A. G. (2007). *Computer Security and Cryptography*: Wiley-Interscience.

(Kopena y Regli, 2003) Kopena, J., Regli, W. C. (2003). DAMLJessKB: A Tool for Reasoning with the Semantic Web. *IEEE Intelligent Systems, 18*(3), 74-77. doi: 10.1109/MIS.2003.1200733

(Kosuga, *et al.*, 2007) Kosuga, Y., Kono, K., Hanaoka, M., Hishiyama, M., Takahama, Y. (2007). Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection. In *23rd Annual Computer Security Applications Conference* (pp. 107-117). Miami Beach, Florida, USA IEEE Computer Society. doi:10.1109/ACSAC.2007.37

(Krüegel, *et al.*, 2002) Krüegel, C., Toth, T., Kirda, E. (2002). Service specific anomaly detection for network intrusion detection. In *ACM symposium on Applied computing (SAC '02)* (pp. 201-208). Madrid, Spain: ACM. doi:10.1145/508791.508835

(Krüegel y Vigna, 2003) Krüegel, C., Vigna, G. (2003). Anomaly detection of web-based attacks. In *10th ACM conference on Computer and communications security (CCS '03)* (pp. 251-261). Washington D.C., USA: ACM. doi:10.1145/948109.948144

(Krutz y Vines, 2002) Krutz, R. L., Vines, R. D. (2002). *The CISSP Prep Guide: Gold Edition*. New York, NY, USA: John Wiley & Sons, Inc.

(Kumar y NandaMohan, 2008) Kumar, K. S. A., NandaMohan, D. V. (2008). Novel Anomaly Intrusion Detection Using Neuro-Fuzzy Inference System. *IJCSNS International Journal of Computer Science and Network Security, 8*(8), 6-11.

(Lawrence*, et al.*, 2004) Lawrence, K., Kaler, C., Nadalin, A., Monzillo, R., Hallam-Baker, P. (2004). *Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)*. Organization for the Advancement of Structured Information Standards (OASIS). Retrieved from http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

(Laza*, et al.*, 2003) Laza, R., Pavón, R., Corchado, J. M. (2003). A Reasoning Model for CBR_BDI Agents Using an Adaptable Fuzzy Inference System. In Conejo, R., Urretavizcaya, M.,De-la Cruz, J.-L. P. (Eds.), *10th Conference of the Spanish Association for Artificial Intelligence (CAEPIA-TTIA03)* (Vol. 3040, pp. 96-106): Springer. doi:10.1007/b98369

(Lazarevic*, et al.*, 2005) Lazarevic, A., Srivastava, J., Kumar, V. (2005). A Survey of Intrusion Detection techniques. In *Managing Cyber Threats: Issues, Approaches and Challenges*: Kluwer Academic Publishers.

(Leake, 1998) Leake, D. (1998). Cognition as case-based reasoning. In Bechtel, W.,Graham, G. (Eds.), *A Companion to Cognitive Science* (pp. 465–476). Blackwell, Oxford

(Leake*, et al.*, 1996) Leake, D., Kinley, A., Wilson, D. (1996). Linking Adaptation and Similarity Learning. In *18th Annual Conference of the Cognitive Science Society*. New Jersey: Lawrence Erlbaum.

(Leake*, et al.*, 2000) Leake, D. B., Bauer, T., Maguitman, A., Wilson, D. C. (2000). Capture, storage and reuse of lessons about information resources: Supporting task-based information search. In *AAAI-00 Workshop on Intelligent Lessons Learned Systems* (pp. 33-37). Austin, Texas: AAAI Press.

(LeCun*, et al.*, 1998) LeCun, Y., Bottou, L., Orr, G. B., Müller, K. R. (1998). Efficient BackProp. In Heidelberg, S. B. (Ed.), *Neural Networks: Tricks of the Trade* (Vol. 1524/1998, pp. 546). doi:10.1007/3-540-49430-8

(Lemos, 2009) Lemos, R. (2009). *Twitter, Facebook fend off DoS attacks*. Retrieved from http://www.securityfocus.com/brief/992

(Li, 2004) Li, W. (2004). Using Genetic Algorithm for network intrusion detection. In *United States Department of Energy Cyber Security Group 2004 Training Conference* (pp. 24--27). Kansas City, Kansas. doi:10.1.1.89.3125

(Limeback, 2008) Limeback, R. (2008). *Simply SQL*. Collingwood, Vic: Sitepoint.

(Lin, 1994) Lin, T. Y. (1994). Fuzzy Patterns in Data Anomaly Detection. In *17th National Computer Security Conference* (pp. 566-580). Baltimore, MD

(Litchfield, 2005) Litchfield, D. (2005). *Data Mining with SQL Injection and Inference*. NGS Software. Retrieved from http://www.databasesecurity.com/webapps/sqlinference.pdf

(Litchfield*, et al.*, 2005) Litchfield, D., Anley, C., Heasman, J., Grindlay, B. (2005). *The Database Hacker's Handbook: Defending Database Servers*: John Wiley & Sons.

(Loh, *et al.*, 2006) Loh, Y.-S., Yau, W.-C., Wong, C.-T., Ho, W.-C. (2006). Design and Implementation of an XML Firewall. In *International Conference on Computational Intelligence and Security* (Vol. 2, pp. 1147-1150). doi:10.1109/ICCIAS.2006.295443

(López De Mántaras, 2001) López De Mántaras, R. (2001). Case-Based Reasoning. In *Machine Learning and Its Applications* (pp. 127-145)

(Lopez De Mantaras, *et al.*, 2005) Lopez De Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M. L., Cox, M. T., Forbus, K., Keane, M., Aamodt, A., Watson, I. (2005). Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review, 20*(3), 215-240. doi: 10.1017/S0269888906000646

(López De Mantaras y Plaza, 1997) López De Mantaras, R., Plaza, E. (1997). Case-Based Reasoning: An Overview. *AI Communications, 10*(1), 21-29.

(Low, *et al.*, 2002) Low, W. L., Lee, J., Teoh, P. (2002). DIDAFIT: Detecting Intrusions in Databases Through Fingerprinting Transactions. In *5th International Conference on Enterprise Information Systems (ICEIS'03)* (pp. 121-128). Ciudad Real - Spain

(Lunt, 1990) Lunt, T. F. (1990). IDES: an intelligent system for detecting intruders. In *Symposium: Computer Security, Threat and Countermeasures*. Rome, Italy.

(Lunt y Jagannathan, 1988) Lunt, T. F., Jagannathan, R. (1988). A Prototype Real-Time Intrusion-Detection Expert System. *Security and Privacy, IEEE Symposium on, 0*, 59. doi: 10.1109/SECPRI.1988.8098

(Maes, 1994) Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM, 37*(7), 30-40. doi: 10.1145/176789.176792

(Martin, *et al.*, 2005) Martin, M., Livshits, B., Lam, M. S. (2005). Finding application errors and security flaws using PQL: a program query language. In *20th annual ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications (OOPSLA '05)* (pp. 365-383). San Diego, CA, USA: ACM. doi:10.1145/1094811.1094840

(Martín, *et al.*, 2007) Martín, Q., Cabero, M., De Paz, Y. (2007). *Statistical treatment of data with SPSS. Resolved and commented practices*: Thomson.

(Mas, 2005) Mas, A. (2005). *Agentes Software y Sistemas Multi-Agente. Conceptos, arquitecturas y aplicaciones*: Pearson Educaciónn, S. A. Madrid.

(McClure y Krüger, 2005) McClure, R. A., Krüger, I. H. (2005). SQL DOM: compile time checking of dynamic SQL statements. In *27th international conference on Software engineering (ICSE '05)* (pp. 88-96). St. Louis, MO, USA: ACM. doi:10.1145/1062455.1062487

(Me, 1998) Me, L. (1998). Gassata, a genetic algorithm as an alternative tool for security audit trails analysis. In *First international workshop on the Recent Advances in Intrusion Detection (RAID '98)*

(Michie, *et al.*, 1994) Michie, D., Spiegelhalter, D. J., Taylor, C. C., Campbell, J. (1994). *Machine learning, neural and statistical classification*. Upper Saddle River, NJ, USA: Ellis Horwood.

(Middleton, 2001) Middleton, S. E. (2001). *Interface agents: A review of the field*. University of Southampton. Retrieved from http://eprints.ecs.soton.ac.uk/6280/

(Miller, *et al.*, 2003) Miller, P., Mill, J., Inoue, A. (2003). Synergistic and Perceptual Intrusion Detection and Reinforcement (SPIDER). In *14th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS2003)* (pp. 102-108). Cincinnati OH

(Minton, 1990) Minton, S. (1990). Qualitative Results Concerning the Utility of Explanation-Based Learning. *Artificial Intelligence, 42*, 363-391.

(Moradian y Håkansson, 2006) Moradian, E., Håkansson, A. (2006). Possible attacks on XML Web Services. *International Journal of Computer Science and Network Security (IJCSNS), 6*(1B), 154-170.

(Moujahid, *et al.*, 2004) Moujahid, A., Inza, I., Larranaga, P. (2004). *Algoritmos Genéticos*. Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad del País Vasco. Retrieved from http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/temageneticos.pdf

(Mukkamala, *et al.*, 2002) Mukkamala, S., Janoski, G., Sung, A. (2002). Intrusion detection using neural networks and support vector machines. In *2002 International Joint Conference on Neural Networks (IJCNN '02)* (Vol. 2, pp. 1702-1707). Honolulu, HI , USA: IEEE press.

(Mukkamala, *et al.*, 2005) Mukkamala, S., Sung, A. H., Abraham, A. (2005). Intrusion detection using an ensemble of intelligent paradigms. *Journal of Network and Computer Applications, 28*(2), 167-182. doi: 10.1016/j.jnca.2004.01.003

(Müller, 1996) Müller, J. P. (1996). *The Design of Intelligent Agents: A Layered Approach* (Vol. 1177). Secaucus, NJ, USA: Springer-Verlag New York, Inc.

(Müller, *et al.*, 1994) Müller, J. P., Pischel, M., Thiel, M. (1994). A pragmatic approach to modelling autonomous interacting systems. In *Workshop on Agent Theories, Architectures, and Languages* (pp. 226-240)

(Muthuprasanna, *et al.*, 2006) Muthuprasanna, M., Wei, K., Kothari, S. (2006). Eliminating SQL Injection Attacks - A Transparent Defense Mechanism. In *Eighth IEEE International Symposium on Web Site Evolution (WSE '06)* (pp. 22-32). Washington, DC, USA: IEEE Computer Society. doi:10.1109/WSE.2006.9

(Newcomer y Lomow, 2004) Newcomer, E., Lomow, G. (2004). *Understanding SOA with Web Services*: Addison Wesley Professional.

(Nguyen-Tuong, *et al.*, 2005) Nguyen-Tuong, A., Guarnieri, S., Greene, D., Shirley, J., Evans, D. (2005). Automatically Hardening Web Applications Using Precise Tainting. In *20th IFIP International Information Security Conference (SEC'2005)* (pp. 295-308)

(Noel, *et al.*, 2002) Noel, S., Wijesekera, D., Youman, C. (2002). Modern intrusion detection, data mining, and degrees of attack guilt. In Barbará, D.,Jajodia, S. (Eds.), *Applications of Data Mining in Computer Security* (pp. 2--25): Kluwer Academic Publishers.

(Nwana, 1996) Nwana, H. S. (1996). Software Agents: An Overview. *Knowledge Engineering Review, 11*(3), 205-244.

(OASIS, 2009) OASIS. (2009). *Organization for the Advancement of Structured Information Standards (OASIS)*. Retrieved from http://www.oasis-open.org/home/index.php

(Odell y Huget, 2003) Odell, J., Huget, M. P. (2003). *FIPA Modeling: Interaction Diagrams*. Retrieved from http://www.auml.org/auml/documents/ID-03-07-02.pdf

(Orallo, *et al.*, 2004) Orallo, J. H., Quintana, M. J. R., Ramírez, C. F. (2004). *Introducción a la Minería de Datos*: Pearson Educación.

(Orfila, *et al.*, 2008) Orfila, A., Carbó, J., Ribagorda, A. (2008). Autonomous decision on intrusion detection with trained BDI agents. *Computer Communications, 31*(9), 1803-1813. doi: 10.1016/j.comcom.2007.11.018

(Ossowski y García-Serrano, 1998) Ossowski, S., García-Serrano, A. (1998). Social Co-ordination among Autonomous Problem-Solving Agents. In *Workshops on Commonsense Reasoning, Intelligent Agents, and Distributed Artificial Intelligence* (pp. 134-148). London, UK: Springer-Verlag.

(Padmanabhuni, *et al.*, 2006) Padmanabhuni, S., Singh, V., Kumar, K. M. S., Chatterjee, A. (2006). Preventing Service Oriented Denial of Service (PreSODoS): A Proposed Approach. In *IEEE International Conference on Web Services (ICWS'06)* (pp. 577-584). Washington, DC, USA: IEEE Computer Society. doi:10.1109/ICWS.2006.102

(Pan, *et al.*, 2003) Pan, Z.-S., Chen, S.-C., Hu, G.-B., Zhang, D.-G. (2003). Hybrid neural network and C4.5 for misuse detection. In *Second International Conference on Machine Learning and Cybernetics* (Vol. 4, pp. 2463 - 2467). doi:10.1109/ICMLC.2003.1259925

(Patrick, 2009) Patrick, J. J. (2009). *SQL Fundamentals, SQL Fundamentals* (Third Edition ed.). 501 Boylston Street, Suite 900: Prentice Hall PTR.

(Pietraszek y Berghe, 2005) Pietraszek, T., Berghe, C. V. (2005). Defending against Injection Attacks through Context-Sensitive String Evaluation. In *Recent Advances in Intrusion Detection 2005 (RAID)*. Seattle, Washington, USA: Springer Verlag. doi:10.1007/11663812_7

(Pinzón, *et al.*, 2009a) Pinzón, C., De Paz, J. F., Bajo, J., Corchado, J. M. (2009a). A Multiagent Solution to Adaptively Classify SOAP Message and Protect against DoS Attack. In *Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA'09)* (pp. 181-190). doi:10.1007/978-3-642-14264-2_19

(Pinzón, *et al.*, 2009b) Pinzón, C., De Paz, J. F., Rodríguez, S., Bajo, J., Corchado, J. M. (2009b). A Hybrid Agent-based Classification Mechanism to Detect Denial of Service Attacks. *Journal of Physical Agents, 3*(3), 11-18.

(Pinzón, *et al.*, 2008a) Pinzón, C., De Paz, Y., Bajo, J. (2008a). A Multiagent Based Strategy for Detecting Attacks in Databases in a Distributed Mode. In Corchado, J. M., Rodríguez, S., Llinas, J.,Molina, J. M. (Eds.), *International Symposium on Distributed Computing and Artificial Intelligence (DCAI'8)*

(Vol. 50, pp. 180-188). Salamanca, Spain: Springer Berlin / Heidelberg. doi:10.1007/978-3-540-85863-8

(Pinzón, *et al.*, 2008b) Pinzón, C., De Paz, Y., Cano, R. (2008b). Classification Agent-Based Techniques for Detecting Intrusions in Databases. In Corchado, E., Abraham, A.,Pedrycz, W. (Eds.), *Hybrid Artificial Intelligence Systems (HAIS2008)* (Vol. 5271/2008, pp. 46-53). Salamanca, Spain: Springer Berlin / Heidelberg. doi:10.1007/978-3-540-87656-4_7

(Pinzón, *et al.*, 2009c) Pinzón, C., De Paz, Y., Cano, R., Rubio, M. P. (2009c). An Attack Detection Mechanism Based on a Distributed Hierarchical Multi-agent Architecture for Protecting Databases. In Demazeau, Y., Pavón, J., Corchado, J.,Bajo, J. (Eds.), *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS'09)* (Vol. 55/2009, pp. 246-255). Salamanca, Spain. doi:10.1007/978-3-642-00487-2_26

(Platon, *et al.*, 2007) Platon, E., Mamei, M., Sabouret, N., Honiden, S., Parunak, H. V. (2007). Mechanisms for environments in multi-agent systems: Survey and opportunities. *Autonomous Agents and Multi-Agent Systems, 14*(1), 31-47. doi: 10.1007/s10458-006-9000-7

(Pokahr, *et al.*, 2003) Pokahr, A., Braubach, L., Lamersdorf, W. (2003). Jadex: Implementing a BDI-Infrastructure for JADE Agents. *EXP - in search of innovation (Special Issue on JADE), 3* (3), 76-85

(Porter, *et al.*, 1990) Porter, B. W., Bareiss, R., Holte, R. C. (1990). Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence, 45*(1), 229-263. doi: 10.1016/0004-3702(90)90041-W

(Portnoy, *et al.*, 2001) Portnoy, L., Eskin, E., Stolfo, S. J. (2001). Intrusion Detection with unlabeled data using clustering. In *ACM CSS Workshop on Data Mining Applied to Security* (pp. 333-342). Philadelphia, USA. doi:10.1.1.13.7523

(Pulier y Taylor, 2005) Pulier, E., Taylor, H. (2005). *Understanding Enterprise SOA*. Greenwich, CT, USA: Manning Publications Co.

(Quinlan, 1986) Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning, 1*(1), 81-106. doi: 10.1023/A:1022643204877

(Racine y Yang, 1997) Racine, K., Yang, Q. (1997). Maintaining Unstructured Case Base. In *The Second International Conference on Case-Based Reasoning Research and Development (ICCBR '97)* (pp. 553-564). London, UK: Springer-Verlag.

(Rain Forest, 1998) Rain Forest, P. (1998). *NT Web Technology Vulnerabilities*. Phrack Magazine. Retrieved from http://www.phrack.org/issues.html?issue=54&id=8#article

(Ramadas, *et al.*, 2003) Ramadas, M., Ostermann, S., Tjaden, B. (2003). Detecting Anomalous Network Traffic with Self-organizing Maps. In *Recent Advances in Intrusion Detection* (pp. 36-54): Springer Verlag.

(Rao y Georgeff, 1998) Rao, A. S., Georgeff, M. P. (1998). Decision Procedures for BDI Logics. *Journal of Logic and Computation, 8*(3), 293-343. doi: 10.1093/logcom/8.3.293

(Rao y Georgeff, 1995) Rao, A. S., Georgeff, M. P. (1995). BDI Agents: From Theory to Practice. In *First International Conference on Multiagent Systems*. San Francisco, California: AAAI Press.

(Rao y Georgeff, 1991) Rao, A. S., Georgeff, M. P. (1991). Modeling Rational Agents within a BDI-Architecture. In Allen, J., Fikes, R.,Sandewall, E. (Eds.), *2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)* (pp. 473-484). San Mateo, CA, USA: Morgan Kaufmann publishers Inc. doi:10.1.1.51.5675

(Reilly y Stillman, 1998) Reilly, M., Stillman, M. (1998). Open Infrastructure for Scalable Intrusion Detection. In *IEEE Information Technology Conference* (pp. 129-133)

(Rescorla y Schiffman, 1999) Rescorla, E., Schiffman, A. (1999). *The Secure HyperText Transfer Protocol*. Retrieved from http://www.rfc-editor.org/rfc/rfc2660.txt

(Riesbeck y Schank, 1989) Riesbeck, C. K., Schank, R. C. (1989). *Inside Case-Based Reasoning*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc.

(Robertson*, et al.*, 2006) Robertson, W., Vigna, G., Kruegel, C., Kemmerer, R. A. (2006). Using Generalization and Characterization Techniques in the Anomaly-Based Detection of Web Attacks. In *13th Annual Network and Distributed System Security Symposium (NDSS'2006)*. San Diego, CA, USA

(Rosenberg y Remy, 2004) Rosenberg, J., Remy, D. (2004). *Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption*: Pearson Higher Education.

(Russel y Norvig, 1995) Russel, S., Norvig, P. (1995). *Artificial Intelligence: a modern approach*. New York: Prentice Hall.

(Sadkhan, 2009) Sadkhan, S. B. (2009). On Artificial Intelligence Approaches for Network Intrusion Detection Systems. *MASAUM Journal of Computing, 1*(2), 236-243.

(Scott y Sharp, 2002) Scott, D., Sharp, R. (2002). Abstracting application-level web security. In *11th international conference on World Wide Web (WWW '2002)* (pp. 396-407). Honolulu, Hawaii, USA: ACM. doi:10.1145/511446.511498

(Schank, 1983) Schank, R. C. (1983). *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. New York, NY, USA: Cambridge University Press.

(SchneeweiB, 1995) SchneeweiB, C. (1995). Hierarchical structures in organisations: A conceptual framework. *European Journal of Operational Research, 86*(1), 4-31. doi: 10.1016/0377-2217(95)00058-X

(Schuba*, et al.*, 1997) Schuba, C. L., Krsul, I. V., Kuhn, M. G., spafford, E. H., Sundaram, A., Zamboni, D. (1997). Analysis of a Denial of Service Attack on TCP. In *IEEE Symposium on Security and Privacy (SP '97)* (pp. 208). Washington, DC, USA: IEEE Computer Society.

(Sekar, 2009) Sekar, R. (2009). An Efficient Black-box Technique for Defeating Web Application Attacks. In *16th Annual Network & Distributed System Security (NDSS'09)*. San Diego, CA

(Shah, *et al.*, 2003) Shah, H., Undercoffer, J., Joshi, A. (2003). Fuzzy Clustering for Intrusion Detection. In *12th IEEE International Conference on Fuzzy Systems* (pp. 1274 - 1278). doi:10.1109/FUZZ.2003.1206525

(Shahriar y Zulkernine, 2008) Shahriar, H., Zulkernine, M. (2008). MUSIC: Mutation-based SQL Injection Vulnerability Checking. In *The Eighth International Conference on Quality Software (QSIC '08)* (pp. 77-86). Washington, DC, USA: IEEE Computer Society. doi:10.1109/QSIC.2008.33

(Shin, *et al.*, 2006) Shin, Y., Williams, L., Xie, T. (2006). *SQLUnitGen: Test Case Generation for SQL Injection Detection*. North Carolina State University, Department of Computer Science. Retrieved from http://www.csc.ncsu.edu/faculty/xie/publications/TR-2006-21.pdf

(Shiu y Pal, 2004) Shiu, S., Pal, S. K. (2004). *Foundations of Soft Case-Based Reasoning*: John Wiley & Sons.

(Sierra, 2006) Sierra, B. (2006). *Aprendizaje Automático: Conceptos Básicos y Avanzados: Aspectos Prácticos Utilizando el Software WEKA* (1 ed.). Madrid, España: Prentice Hall.

(Sierra, *et al.*, 2001) Sierra, C., López De Mántaras, R., Busquets, D. (2001). Multiagent Bidding Mechanisms for Robot Qualitative Navigation. In *7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages (ATAL'01)* (pp. 198-212). London, UK: Springer-Verlag.

(Singh, 2010) Singh, I. (2010). *tcpmon: An open-source utility to Monitor A TCP Connection*. Retrieved from https://tcpmon.dev.java.net/

(Singhal, 2007) Singhal, A. (2007). Web Services Security: Challenges and Techniques. In *Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY '07)* (pp. 282-282). Washington, DC, USA: IEEE Computer Society. doi:10.1109/POLICY.2007.50

(Singhal, *et al.*, 2007) Singhal, A., Winograd, T., Scarfone, K. (2007). *NIST Special Publication 800-95. Guide to Secure Web Services*. Retrieved from http://csrc.nist.gov/publications/nistpubs/800-95/SP800-95.pdf

(Skaruz y Seredynski, 2007) Skaruz, J., Seredynski, F. (2007). Recurrent neural networks towards detection of SQL attacks. In *IEEE International: Parallel and Distributed Processing Symposium (IPDPS 2007)* (pp. 1-8). Long Beach, CA, USA doi:10.1109/IPDPS.2007.370428

(Smaha, 1988) Smaha, S. (1988). Haystack: An intrusion detection system. In *Fourth Aerospace Computer Security Applications Conference* (pp. 37-44). Austin, Texas. doi:10.1109/ACSAC.1988.113412

(Smith y Medin, 1981) Smith, E., Medin, D. (1981). *Categories and Concepts*: Harvard University Press.

(Smyth y Keane, 1995) Smyth, B., Keane, M. T. (1995). Remembering To Forget: A Competence-Preserving Case Deletion Policy for Case-Based Reasoning Systems. In *13th International Joint Conference on Artificial Intelligence* (pp. 377-382). Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc.

(Spafford y Zamboni, 2000) Spafford, E. H., Zamboni, D. (2000). Intrusion detection using autonomous agents. *The International Journal of Computer and Telecommunications Networking, 34*(4), 547-570. doi: 10.1016/S1389-1286(00)00136-5

(Srivatsa*, et al.*, 2008) Srivatsa, M., Iyengar, A., Yin, J., Liu, L. (2008). Mitigating application-level denial of service attacks on Web servers: A client-transparent approach. *ACM Transactions on the Web 2*(3), 1-49. doi: 10.1145/1377488.1377489

(Stamp, 2006) Stamp, M. (2006). *Information Security: Principles and Practice*: Wiley InterScience.

(Stanfill y Waltz, 1988) Stanfill, C., Waltz, D. L. (1988). The Memory-Based Reasoning Paradigm? In *Proceeding Case-Based Reasoning Workshop* (pp. 414-424). Clearwater Beach, FL

(Stolfo*, et al.*, 1997) Stolfo, S., Tselepis, A. L. P. S., Prodromidis, A. L., Tselepis, S., Lee, W., Fan, D. W., Chan, P. K. (1997). JAM: Java Agents for Meta-Learning over Distributed Databases. In *3rd International Conference Knowledge Discovery and Data Mining* (pp. 74-81): AAAI Press.

(Su y Wassermann, 2006) Su, Z., Wassermann, G. (2006). The Essence of Command Injection Attacks in Web Applications. In *33rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL'06)* (pp. 372-382). Charleston, South Carolina, USA. doi:10.1145/1111037.1111070

(Sycara*, et al.*, 2003) Sycara, K., Paolucci, M., Van Velsen, M., Giampapa, J. (2003). The RETSINA MAS Infrastructure. *Autonomous Agents and Multi-Agent Systems, 7*(1-2), 29-48. doi: 10.1023/A:1024172719965

(Tajbakhsh*, et al.*, 2009) Tajbakhsh, A., Rahmati, M., Mirzaei, A. (2009). Intrusion detection using fuzzy association rules. *Applied Soft Computing, 9*(2), 462-469. doi: 10.1016/j.asoc.2008.06.001

(Thomas y Williams, 2007) Thomas, S., Williams, L. (2007). Using Automated Fix Generation to Secure SQL Statements. In *Third International Workshop on Software Engineering for Secure Systems (SESS '07)* (pp. 9). Washington, DC, USA: IEEE Computer Society. doi:10.1109/SESS.2007.12

(Tsai*, et al.*, 2009) Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., Lin, W.-Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications, 36*(10), 11994-12000. doi: 10.1016/j.eswa.2009.05.029

(Tuba y Bulatovic, 2009) Tuba, M., Bulatovic, D. (2009). Design of an intrusion detection system based on Bayesian networks. *W. Trans. on Comp., 8*(5), 799-809.

(Valdes y Skinner, 2000) Valdes, A., Skinner, K. (2000). Adaptive, Model-Based Monitoring for Cyber Attack Detection. In *Third International Workshop on Recent Advances in Intrusion Detection (RAID '00)* (pp. 80-92). London, UK: Springer Verlag.

(Valeur*, et al.*, 2005) Valeur, F., Mutz, D., Vigna, G. (2005). A Learning-Based Approach to the Detection of SQL Attacks. In *Conference on Detection of*

Intrusions and Malware and Vulnerability Assessment (DIMVA) (pp. 123-140). Vienna, Austria. doi:10.1.1.94.9201

(Vapnik*, et al.*, 1996) Vapnik, V., Golowich, S. E., Smola, A. (1996). Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing. In Mozer, M., Jordan, M.,Petsche, T. (Eds.), *Advances in Neural Information Processing Systems 9* (pp. 281-287): MIT Press.

(Vapnik, 1995) Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer Verlag New York, Inc.

(Varaiya, 2000) Varaiya, P. (2000). A Question About Hierarchical Systems. In Djaferis, T. E.,Schick, I. C. (Eds.), *System Theory: Modeling, Analysis and Control* (Vol. 518): Kluwer. doi:10.1.1.33.1306

(Veloso y Carbonell, 1993) Veloso, M. M., Carbonell, J. G. (1993). Derivational Analogy in PRODIGY: Automating Case Acquisition, Storage, and Utilization. *Machine Learning, 10*(3), 249-278. doi: 10.1023/A:1022686910523

(Viñuela y León, 2004) Viñuela, P. I., León, I. M. G. (2004). *Redes de Neuronas Artificiales-Un Enfoque Práctico*. Madrid, España: Pearson Educación S.A.

(Vorobiev y Han, 2006) Vorobiev, A., Han, J. (2006). Security Attack Ontology for Web Services. In *Second International Conference on Semantics, Knowledge, and Grid (SKG '06)* (pp. 42-42). Guilin, Guangxi, China IEEE Computer Society. doi:10.1109/SKG.2006.85

(W3C, 2009) W3C. (2009). *World Wide Web Consortium (W3C)*. Retrieved from http://www.w3.org/

(Wang*, et al.*, 2006) Wang, H. Q., Wang, Z. Q., Zhao, Q., Wang, G. F., Zheng, R. J., Liu, D. X. (2006). Mobile agents for network intrusion resistance. In *APWeb Workshops 2006* (Vol. 3842, pp. 965-970). Harbin, China: Springer, Heidelberg. doi:10.1007/11610496_134

(Wang, 2006) Wang, J. (2006). Defending Against Denial of Web Services Using Sessions. In *IEEE/IST Workshop on: Monitoring, Attacking Detection and Mitigation*

(Wassermann*, et al.*, 2007) Wassermann, G., Gould, C., Su, Z., Devanbu, P. (2007). Static Checking of Dynamically Generated Queries in Database Applications. *ACM Transactions on Software Engineering and Methodology, 16*(4), 14. doi: 10.1145/1276933.1276935

(Wassermann y Su, 2004) Wassermann, G., Su, Z. (2004). An Analysis Framework for Security in Web Applications. In *3rd Workshop on Specification and Verification of Component-Based Systems (SAVCBS 2004)* (pp. 70-78). Newport Beach, California

(Weyns*, et al.*, 2004) Weyns, D., Parunak, H. V. D., Michel, F., Holvoet, T., Ferber, J. (2004). Environments for Multiagent Systems State-of-the-Art and Research Challenges. In Weyns, D., Parunak, H. V. D.,Michel, F. (Eds.), *First International Workshop (E4MAS'04)* (pp. 1-47). New York, NY, USA: Springer.

(Witten y Frank, 2000) Witten, I. H., Frank, E. (2000). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation*. San Francisco: Morgan Kaufmann.

(Wooldridge, 2002) Wooldridge, M. (2002). *Introduction to MultiAgent Systems*: John Wiley & Sons.

(Wooldridge y Jennings, 1995) Wooldridge, M., Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review, 10*(2), 115-152.

(WS-I, 2009) WS-I. (2009). *Web Services Interoperability Organization*. Retrieved from http://www.ws-i.org/deliverables/workinggroup.aspx?wg=basicprofile

(Wu y Banzhaf, 2010) Wu, S. X., Banzhaf, W. (2010). The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing, 10*(1), 1-35. doi: 10.1016/j.asoc.2009.06.019

(Xin*, et al.*, 2003) Xin, J., Dickerson, J. E., Dickerson, J. A. (2003). Fuzzy feature extraction and visualization for intrusion detection. In (Vol. 2, pp. 1249-1254). doi:10.1109/FUZZ.2003.1206610

(Xin WEI y qing WU, 2008) Xin WEI, Y., qing WU, M. (2008). KFDA and clustering based multiclass SVM for intrusion detection. *The Journal of China Universities of Posts and Telecommunications, 15*(1), 123-128. doi: 10.1016/S1005-8885(08)60074-6

(Ye y Li., 2000) Ye, N., Li., X. (2000). Application of Decision Tree Classifier to Intrusion Detection. In University, C. (Ed.), *Second International Conference on DATA MINING 2000*. Cambridge University, UK

(Ye, 2008) Ye, X. (2008). Countering DDoS and XDoS Attacks against Web Services. In *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing* (Vol. 1 pp. 346-352). Washington, DC, USA: IEEE Computer Society. doi:10.1109/EUC.2008.61

(Yee*, et al.*, 2007) Yee, C. G., Shin, W. H., Rao, G. S. V. R. K. (2007). An Adaptive Intrusion Detection and Prevention (ID/IP) Framework for Web Services. In *International Conference on Convergence Information Technology (ICCIT '07)* (pp. 528-534). Washington, DC, USA: IEEE Computer Society.

(Zadeh, 1965) Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control, 8*(3), 338-353.

(Zhang*, et al.*, 2003) Zhang, J., Ding, Y., Gong, J. (2003). Intrusion Detection System Based on Fuzzy Default Logic. In *12th IEEE International Conference on Fuzzy Systems (FUZZ'03)* (Vol. 2, pp. 1350-1356)

(Zhang*, et al.*, 2001) Zhang, R., Qian, D., Bao, C., Wu, W., Guo, X. (2001). A Multi-Agent based Intrusion Detection Architecture. In *International Conference on Computer Networks and Mobile Computing* (Vol. 0, pp. 494-501): IEEE Computer Society. doi:10.1109/ICCNMC.2001.962638

(Zhang*, et al.*, 2009) Zhang, Y., Ye, X., Xie, F., Peng, Y. (2009). A Practical Database Intrusion Detection System Framework. In *Ninth IEEE International Conference on Computer and Information Technology (CIT'2009)* (pp. 342-347). Washington, DC, USA: IEEE Computer Society. doi:10.1109/CIT.2009.69

(Zhao*, et al.*, 2007) Zhao, K., Yang, K., Zhang, M., Wang, J., Hu, L. (2007). Denial of Service Attack Simulation Based-on CASL. In *IEEE International*

Workshop on Anti-counterfeiting, Security, Identification (pp. 266-269). Xiamen, Fujian doi:10.1109/IWASID.2007.373741

(Zhi-Wei*, et al.*, 2005) Zhi-Wei, N., Yu, L., Feng-Gang, L., Shan-Lin, Y. (2005). Case base maintenance based on outlier data mining. In *International Conference on Machine Learning and Cybernetics* (Vol. 5, pp. 2861- 2864). Guangzhou, China doi:10.1109/ICMLC.2005.1527430