# Redes neuronales aplicadas al proceso de aprendizaje de un sistema de respuestas a intrusiones automático

Pilar Holgado, Víctor A. Villagra, Verónica Mateos.
Departamento de Ingeniería y Sistemas Telemáticos,
Universidad Politécnica de Madrid
Avenida Complutense, 30, 28040, Madrid
pilarholgado@dit.upm.es, villagra@dit.upm.es, vmateos@dit.upm.es.

***Resumen*- La contribución de este artículo es el uso de métodos de aprendizaje automático en la arquitectura realizada dentro del proyecto RECLAMO en trabajos previos. La arquitectura se basa en un AIRS (sistema de respuestas a intrusiones automático) que infiere la respuesta más apropiada a un ataque, teniendo en cuenta el tipo de ataque, la información de contexto del sistema y la red, y la reputación del IDS que ha reportado la alerta. También, es imprescindible conocer el ratio de éxito y fracaso de las respuestas lanzadas ante un ataque, de tal manera que, además de tener un sistema adaptativo, se consiga la capacidad de autoaprendizaje. En este ámbito es donde las redes neuronales entran en juego, aportando la clasificación de éxito/fracaso de las respuestas.**

***Palabras Clave*- Métodos de aprendizaje automático, Redes Neuronales, Retropropagación, Ontología, OWL, IDMEF, AIRS.**

## I. INTRODUCCIÓN

Todos los sistemas que componen una unidad organizativa tienen que estar protegidos contra ataques externos, ya sea para que no decaigan los niveles de disponibilidad de los servicios que ofrecen a sus clientes, como para mantener información confidencial de la empresa y el funcionamiento correcto de sus aplicaciones.

Cada vez es mayor el número de eventos de seguridad, su sofisticación y extensión [1]. Los Sistemas de Detección de Intrusiones (IDS) [2], han evolucionado rápidamente, y en la actualidad, hay herramientas muy sofisticadas basadas en los distintos paradigmas (estadísticas basadas en anomalía [3], basadas en firmas e hibridas [4]) con un alto nivel de confiabilidad. Los IPS (Sistemas de Prevención de Intrusiones) también se han desarrollado por combinación de un IDS con respuestas reactivas básicas, como reestablecer una conexión. Los IRS (Sistemas de Respuestas a Intrusiones) aprovechan el concepto de IPS para logar una respuesta específica de acuerdo a unas reglas predefinidas.

Hoy en día, los IRSs están jugando un papel importante en la arquitectura de seguridad. Estos sistemas mitigan el impacto de ataques que intentan comprometer la integridad, confidencialidad y disponibilidad de los recursos del sistema. Un Sistema de Respuestas Automático (AIRS) proporciona la mejor defensa posible y acorta o cierra la ventana de oportunidades hasta que el administrador del sistema puede tomar un rol activo en defender contra el ataque.

Un AIRS es un sistema de seguridad que elige y ejecuta respuestas automatizadas contra las alertas detectadas por IDSs, con el objetivo de mitigarlas o reducir su impacto [5].

En el proceso de respuestas a intrusiones es necesario definir varias métricas que ofrecen un medio para medir diferentes parámetros útiles para la selección de la respuesta, tales como, confiabilidad del IDS, el nivel de actividad de la red, fiabilidad de los informes de intrusión, la importancia de los componentes de la red y la complejidad, gravedad, coste y eficiencia de las respuestas.

Los AIRS existentes tienen un enfoque fijo para las métricas de respuesta, por lo que las métricas no pueden ser elegidas de manera dinámica. En este artículo se propone una arquitectura de seguridad que es capaz de seleccionar la respuesta más apropiada dinámicamente teniendo en cuenta una serie de factores, como el contexto del sistema, el coste de la respuesta, el valor del recurso atacado y la eficiencia de las respuestas.

En este contexto, el proyecto RECLAMO (Red de sistemas de Engaño virtuales y Colaborativos basados en sistemas Autónomos de respuesta a intrusiones y Modelos de cOnfianza), dentro del proyecto R&D financiado por el Ministerio de Ciencia e Innovación, define un AIRS capaz de interpretar métricas dinámicamente.

El sistema autónomo desarrollado está basado en modelos formales de información definidos con ontologías [7] para combinar la información de las intrusiones, parámetros de autoevaluación del sistema aprendidos en usos anteriores, confianza y reputación de los diferentes componentes, así como la información enviada por otros IDS/IPS colaborativos en el mismo o distinto dominio. Esta información será evaluada con un conjunto de métricas de seguridad representadas en un lenguaje formal de especificación de comportamiento, SWRL (Semantic Web Rule Language) [6], para razonar e inferir la respuesta más apropiada, teniendo en cuenta toda la información de entrada y otros criterios especificados.

Para homogeneizar la información se usan ontologías basadas en IDMEF (Intrusion Detection Message Exchange Format) [8], donde las alertas son representadas como instancias de clases en la ontología [9]. La ontología ha sido definida usando OWL (Web Ontology Language) [10], aprovechando las ventajas de la web semántica, como es la inferencia.

En definitiva, se ha desarrollado un AIRS adaptativo, haciendo un balance del daño que causaría la intrusión con el coste de la respuesta. Además, se realiza el análisis de la eficiencia de la respuesta, para que el AIRS sea totalmente autónomo y aprenda de manera automática.

Para la evaluación de la respuesta que se ha lanzado tras la llegada de un ataque se propone el uso de redes neuronales. Las redes neuronales se encuentran dentro de una rama de inteligencia artificial denominado aprendizaje automático. Dentro de este grupo, hay una gran cantidad de técnicas de aprendizaje [11]. El papel del algoritmo realizado es la clasificación del éxito o no de la respuesta ejecutada, y mediante un ratio de respuesta, el sistema sea capaz de auto-aprender para futuras incidencias del mismo tipo.

A continuación se indica cómo se va a estructurar el artículo. Primero se muestran trabajos previos de sistemas similares al propuesto. Posteriormente se resumirá la arquitectura del AIRS basado en Ontologías, la métrica y el proceso de inferencia llevado a cabo. Tras ello, se introducen las redes neuronales, el algoritmo seleccionado y su proceso de aprendizaje. Después, se detallará el escenario de pruebas. Y finalmente, las conclusiones y el trabajo futuro.

## II. TRABAJOS PREVIOS

El uso de sistemas automáticos para la prevención, detección y respuesta de intrusiones es un concepto clave en la investigación de los últimos años. A continuación se muestra algunos de estos proyectos actuales.

Un sistema de respuestas a intrusiones adaptativo basado en inmunidad artificial llamado MAIM [16] implementa un modelo de políticas de respuesta automática de acuerdo con el peligro global de la red y el host en tiempo real, en vez de centrarse en los ataques individuales. Para ello utiliza una estrategia basada también en sistemas biológicos como es el sistema inmunológico. Por otro lado, realizan el aprendizaje del estado de peligro de la red para la detección de falsos positivos y luego realizar una respuesta más o menos estricta a un ataque con políticas predefinidas. En nuestro caso, realizamos el análisis del contexto para la detección de falsos positivos y realizamos una respuesta más o menos exigente en función del coste del ataque y la respuesta con respecto a la importancia del recurso comprometido. Además, el aprendizaje se realiza sobre la efectividad de la respuesta a un ataque dado.

En [17], se aborda las intrusiones a través de varias ejecuciones con ráfagas de respuesta y al final de cada ráfaga, un mecanismo para medir la efectividad de las respuestas. Esto se realiza con una máquina de estados finito. La efectividad de la respuesta se mide en función de las consecuencias sobre el host relativas a la integridad, disponibilidad, confidencialidad y rendimiento del mismo, lo que repercute en el orden en que serán ejecutadas dentro de la ráfaga. Por otro lado, divide en niveles las ráfagas de respuestas, basado en la efectividad de estas en el histórico de respuestas aplicadas, de manera que modifica el orden del conjunto de respuestas a ejecutar. En nuestro caso, la efectividad medida en las respuestas ejecutadas es totalmente adaptativo y cuantitativo, de manera que es más precisa la clasificación de respuestas.

COSIRS [18], se basan en tres factores para la evaluación de las respuestas, el coste del daño causado por la intrusión, el coste de la respuesta automática a la intrusión y el coste operacional. Al contrario que el sistema presentado, sólo se basan en la efectividad de la respuesta anterior por lo que el sistema carece de inteligencia.

En general, las redes neuronales se han utilizado en distintos ámbitos de la seguridad de red. Entre ellos se encuentra el ámbito de detección de intrusiones. Se pueden encontrar tanto el uso de algoritmos supervisados como no supervisados. Por ejemplo en [19] y [20] usan el algoritmo de Retropropagación.

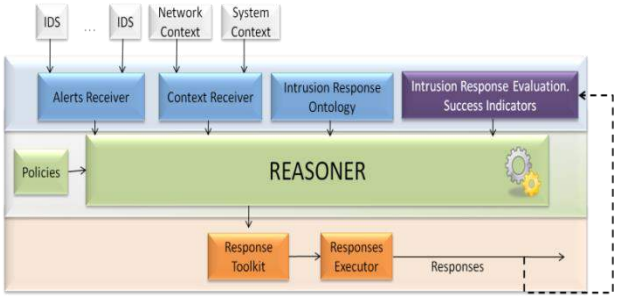


Fig. 1. Arquitectura del AIRS basado en Ontologías

## III. ARQUITECTURA

En la Fig. 1. se representa el funcionamiento del AIRS propuesto en [12]. El objetivo de la arquitectura mostrada es elegir la respuesta óptima de un conjunto de respuestas disponibles.

El AIRS recibe un conjunto de entradas, incluyendo, informes de intrusiones, información de contexto, políticas de las métricas de seguridad y la ontología de respuesta a intrusiones. Las políticas especifican diferentes métricas que serán elegidas dependiendo del contexto y tipo de intrusión. A continuación se explica en detalle los módulos principales.

- *Reasoner*: Ejecuta procesos de inferencia para elegir la mejor respuesta basado en los módulos, *Policies*, *Alerts Receiver*, *Context Receiver* e *Intrusion Response Ontology*. Utiliza OWL para definir toda la información del proceso de respuesta. Tras procesar las entradas se infiere un conjunto de respuestas óptimas en Response Toolkit. Por último, el módulo Response Executor es el que lleva a cabo la respuesta inferida.
- *Policies:* Conjunto de reglas SWRL que especifican el comportamiento del AIRS. Estas políticas son definidas por el administrador del sistema. En este módulo se definen las métricas de respuesta de acuerdo con la información de contexto para así realizar un AIRS adaptativo, proactivo y cost-sensitive.
- *Alerts Receiver*: Tratan con alertas de distintos IDSs, por lo que es común que generen la misma alerta pero con distinto formato y sintaxis; por lo que este módulo garantiza que las alertas sean semánticamente iguales y se correspondan con los conceptos definidos en la Ontología.
- *Context Receiver*: Realiza la correspondencia de los datos de los módulos recogidos en *Network Context* y *System Context* con los conceptos definidos en la Ontología.
- *Intrusion Response Ontology:* Define los conceptos y relaciones necesarios para gestionar la seguridad de redes a través de un AIRS (Fig. 2.). La ontología se basa en la estructura IDMEF, que sigue un modelo de clases y propiedades. Hay dos clases directamente relacionadas con el sistema de evaluación, *Response* y

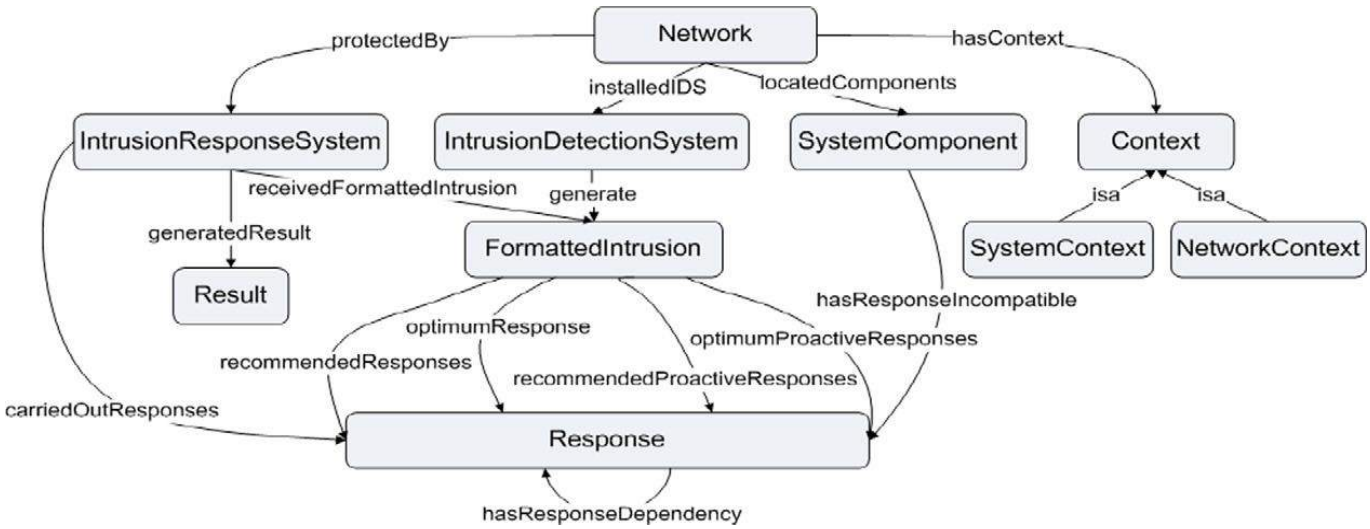


Fig. 2. Ontología de respuestas a intrusiones

*Result.* Durante la integración del sistema de evaluación en el AIRS, será necesario modificar y añadir propiedades a estas clases. En la clase *Response*, *executionTimes* y *sucessFactor*, número de veces que se lanza una respuesta y ratio de satisfacción de la respuesta, respectivamente. En la clase *Result*, se añade la propiedad *responseEfficiency*, que se corresponde con el cociente entre *executionTimes* y *successFactor*. Se actualizan dichos valores en la respuesta cada vez que es lanzada.

- *Intrusion Response Evaluation:* Se encarga de evaluar las respuestas lanzadas por el AIRS mediante el uso de una red neuronal previamente entrenada con muestras del contexto del sistema y de red.

## IV. MÉTRICAS DE RESPUESTA Y PROCESO DE INFERENCIA

Las métricas aplicadas para inferir la respuesta más apropiada han sido definidas y analizadas en [13]. A continuación se muestra un resumen:

- Reducción del daño: Realiza el balance del coste del daño causado por el ataque y el coste de despliegue de la respuesta. Es la primera de las métricas aplicadas.
- Coste mínimo: De todas las respuestas inferidas se lanza la respuesta de menor coste cuando el componente afectado no es muy relevante para la organización.
- Alta severidad y eficiencia: Si el recurso de la organización es muy relevante o crítico se utiliza esta métrica para maximizar la severidad y éxito de la respuesta.

El proceso de inferencia del AIRS se basa en estas métricas para razonar la respuesta más apropiada siguiendo los siguientes pasos:

1. Recolección de información cada vez que llega una intrusión:
    a. Comparación del contexto del sistema y la red en un estado normal, es decir libre de intrusiones y en un entorno seguro, que ha sido almacenado anteriormente con el contexto después de la intrusión
    b. Informes extraídos de los IDSs
2. Inferencia de un conjunto de respuestas recomendadas:
    a. Primero se comprueba si es la primera intrusión que llega al sistema.
    b. En el caso de que sea la primera, se infieren las respuestas recomendadas mediante las políticas SWRL a través de la Métrica de reducción del daño, basada en el tipo de intrusión y los parámetros de contexto.
    c. Si el ataque es similar a uno anterior, se ejecuta la respuesta seleccionada anteriormente si fue satisfactoria.
    d. En otro caso, se infieren las respuestas recomendadas como en b.
3. Se selecciona la respuesta óptima a inferir del conjunto de respuestas recomendadas, de acuerdo con la importancia del recurso comprometido:
    a. Recursos poco relevantes: Se aplica la métrica del coste mínimo
    b. Recursos relevantes: Se aplica la métrica de mayor eficiencia y severidad de menor coste
    c. Recursos críticos: Utiliza la métrica de mayor eficiencia y severidad, aplicando la respuesta más severa sin tener en cuenta el coste de su despliegue.

## V. REDES NEURONALES

Las redes neuronales artificiales son redes interconectadas masivamente en paralelo y con organización jerárquica, las cuales intentan interaccionar con los objetos del mundo real del mismo modo que hace el sistema nervioso [14].

Las neuronas que conforman una red neuronal están interconectadas a través de su sinapsis, que es lo que permite la transmisión de información. No todas las conexiones son iguales, por lo que se le asigna un peso de conexión.

Los pesos obtenidos tras la fase de aprendizaje determinan la salida de la red, por lo que se podría decir que forman la memoria de la red neuronal.

Las redes neuronales son particularmente útiles para solventar problemas que no pueden expresarse como una serie de pasos, tales como reconocimiento de patrones, clasificación, predicción de series y minería de datos. En nuestro caso será utilizado para la clasificación de respuestas a intrusiones.

.

La clasificación es un proceso muy relacionado con el reconocimiento de patrones. Una red neuronal entrenada para clasificación está diseñada para tomar muestras de entradas y clasificarlas en grupos. Estos pueden ser difusos, es decir sin límites claramente definidos; o con fronteras definidas en el caso de seleccionar valores umbrales.

A continuación se van a presentar las ventajas de estas redes:

1. Aprendizaje adaptativo mediante algoritmos de entrenamiento, de acuerdo a experiencias previas
2. Autoorganización: Representación propia de la información.
3. Tolerancia a fallos: Robustez frente a destrucción parcial de la información.
4. Integración modular, por su fácil inserción dentro de la tecnología existente.
5. Son una eficiente herramienta de clasificación que puede ser aplicada a problemas complejos con gran cantidad de parámetros.
6. Es confiable en la predicción de problemas de regresión y clasificación.
7. Proporcionan muy buena clasificación en entornos ruidosos.
8. Robustez frente a destrucción parcial.

Por último, presentamos las desventajas de este tipo de algoritmos:

1. La fiabilidad de las predicciones se degrada con la presencia de múltiples soluciones causadas por muchos mínimos locales.
2. Es difícil determinar y entender su rendimiento.
3. Hay que ajustar cuidadosamente por el usuario un gran número de parámetros para obtener unos buenos resultados.
4. Son muy lentas tanto en fase de entrenamiento como de validación
5. Utilización de validación cruzada para evitar sobreajustes.
6. Sensibles a los conjuntos de datos incompletos
7. La falta de reglas para la definición de la red como por ejemplo la elección del algoritmo de aprendizaje, la arquitectura de la red, el número de neuronas por capas, el número de capas.

### *A. Algoritmos de entrenamiento de las redes neuronales*

Hay muchas maneras de entrenar una red neuronal. Generalmente, las categorías de algoritmos de entrenamiento se encuentran dentro de:

- Supervisado: Se sabe a priori como son las características a clasificar. Al conocer la salida esperada, el entrenamiento se beneficia de la supervisión de un maestro. Por tanto, se necesita un conjunto de entrenamiento con un conjunto de datos de entrada previamente clasificado o con su salida objetivo conocida. La red supervisada realiza una serie épocas, hasta que la salida se corresponde con la esperada con un ratio de error razonablemente bajo o con la condición de parada que se considere más apropiada. Una época se corresponde con la ejecución del algoritmo para todas las muestras del conjunto de entrenamiento. Es el tipo de entrenamiento más habitual. Ejemplos de este tipo de entrenamiento son el Perceptrón Simple, Red Adeline, el Perceptrón Multicapa, red de Retropropagación y Memoria Asociativa bidireccional.
- No supervisado: No se sabe las categorías o características a clasificar, por lo que este aprendizaje se basa en una estadística para determinar el patrón. Se suele utilizar para la clasificación de los patrones en grupos diferenciados que se van descubriendo a medida que progresa el entrenamiento. Al igual que en el aprendizaje supervisado, se ejecuta el algoritmo durante varias épocas hasta llegar a la condición de parada. Ejemplo de este tipo de redes son las Memorias Asociativas, las redes de Hopfield, la Máquina de Boltzmann, la Máquina de Cauchy, las redes de Aprendizaje Competitivo, las redes de Kohoneno, Mapas Autoorganizativos, las Redes de Resonacia Adaptativa (ART).
- Redes híbridas: Tienen un enfoque mixto entre las dos anteriores. Utilizan una función de mejora para facilitar la convergencia. Un ejemplo son las redes de Base Radial.
- Aprendizaje Reforzado: Se sitúa a medio camino entre el entrenamiento supervisado y el autoorganizado. Hay que proporcionar datos de entrada a la red neuronal sin indicar la salida esperada. Sin embargo, para cada salida generada por la red neuronal se dice si es o no correcta para cada una de las entradas, para la realización de la realimentación de la red basada en ensayo-error.

## VI. RETROPROPAGACIÓN

En la evaluación de la respuesta seleccionada por el AIRS, se ha optado por el uso de las redes neuronales porque es un problema en el que no se puede determinar si la respuesta ha sido o no satisfactoria de una manera clara para cualquier sistema o intrusión.

Se propone el algoritmo de Retropropagación (backpropagation) o también llamado Perceptrón Multicapa (LMP), ya que se puede disponer de un conjunto de entrenamiento para tener un aprendizaje supervisado. Los beneficios de éste modo de aprendizaje son:

- Redes que convergen más rápido a la clasificación esperada que un algoritmo no supervisado.
- Aprendizaje guiado por observaciones pasadas: Se llevará a cabo a través de un conjunto de ejemplares clasificados que se obtienen durante la experimentación dentro de un sistema de pruebas.

Esta red es capaz de clasificar a partir del contexto si la respuesta ha sido correcta, lo que se representaría con un 1 o incorrecta, con un –1, en la capa de salida de la red. Para darle más detalle al resultado en vez de una función escalón se toma una salida real para así saber en qué factor la respuesta se ha enfrentado a la intrusión, consiguiendo un intervalo comprendido entre -1 y 1 denominándolo SuccessLevel. Tras ello se calcula la eficiencia de la respuesta con las siguientes fórmulas:

$$SuccessFactor = \sum_{i=0}^{j-1} SuccessLevel_i \qquad (1)$$

$$ResponseEfficiency = \frac{SuccessFactor}{ExecutionTimes} \quad (2)$$

Donde j y ExecutionTimes es el número de veces que se ha ejecutado esa respuesta.

### A. *Parámetros de entrada*

Los parámetros se podrían corresponder con los parámetros obtenidos del contexto del sistema y red tras la ejecución de la respuesta. La desventaja radica en que el valor del contexto que podríamos indicar como "normal", es dependiente del dispositivo o sistema de la organización. Por tanto, para que sea totalmente independiente y que el mismo algoritmo se pueda implantar en todos los host, los parámetros elegidos se corresponden con el grado de anomalía del contexto del sistema y la red tras la ejecución de la respuesta lanzada por el AIRS con respecto al contexto normal.

El contexto de red ha sido recogido con un solo parámetro y para el contexto del sistema se han usado siete parámetros obtenidos tras la monitorización del sistema con Nagios Core Tool [15]:

- Estado: Si el sistema está o no activo.
- Latencia: Tiempo en el que los agentes de Nagios tardan en corresponder al servidor de Nagios.
- Uso de CPU: Anomalías debidas a sobrecarga del sistema.
- Espacio de disco: Para comprobar cambios drásticos en el espacio de disco duro.
- Número de procesos activos: La mayoría de las intrusiones contienen procesos que antes no existían.
- Número de usuarios.
- Número de procesos zombies: Algunos virus aumentan el número de procesos zombies para que sufran una degradación del rendimiento.

El contexto de anomalía se encuentra dentro de un rango de [0-10]. Pero, hay que tener en cuenta que la representación de las entradas en una red neuronal debe ser numérica y normalizada, ya que valores muy altos pueden perjudicar la efectividad del algoritmo. Debido a ello, antes de introducir el contexto como entrada de la red, es necesario dividirla por 10, para que quede en un rango de [0-1].

### B. *Función de transferencia*

La función de activación se corresponde con la definición de una función para normalizar la salida de la red neuronal tras el procesamiento de la información. Se clasifican en tres tipos:

- Función identidad: Mantiene el valor de salida procesado
- Umbral: Función escalón binaria o bipolar.
- Sigmoidal: Normaliza teniendo en cuenta un rango real de salida. También puede ser binaria o bipolar.

En este caso se utilizará una función sigmoidal para determinar el grado de satisfacción de la respuesta ante la intrusión. Por otro lado, nos basaremos en funciones bipolares ya que consiguen antes la estabilización del error. Dentro del grupo de funciones sigmoidales probaremos con dos:

- Función Sigmoide logística bipolar:

$$f(x) = \frac{2}{1 + e^{-x}} - 1 \quad (3)$$

- Función Tangencial Hiperbólica:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (4)$$

### C. *Condición de parada*

Como condición de parada se tiene en cuenta el error cuadrático medio (ECM):

$$ECM = \sum (y_{obj} - y_{en})^2 \quad (5)$$

Donde $y_{obj}$ es la salida deseada y $y_{en}$ es la salida generada por el algoritmo.

Por tanto, el algoritmo intenta encontrar los pesos de las neuronas que minimicen el ECM para dar la clasificación más acertada posible.

### D. *Validación*

Esta última etapa es muy importante porque permite determinar si se requiere de entrenamiento adicional. Para ello se debe tener otro conjunto, denominado conjunto de test, que validará la generalización de la red.

Para evitar sobreajustes, es decir, que haya un sobreaprendizaje de la red debido a demasiada información presentada en el conjunto de entrenamiento, se utiliza el método de validación cruzada. Es decir, que toda la muestra disponible se divide en dos partes, una para el conjunto de entrenamiento y otro para el de test, con ejemplos de todos los tipos de patrones.

### E. *Número de neuronas y número de capas*

La relación entre el número total de parámetros y el número de patrones debe estar entre el 10% o 15% para que haya suficientes patrones para la generalización.

Hay que tener en cuenta la relación entre patrones y parámetros, para la elección del número de neuronas de la capa oculta:

- Muy pocas neuronas en la capa oculta conducirán a un alto error de entrenamiento y un alto error de generalización debido al underfitting.
- Si por el contrario, se tienen muchas neuronas en la capa oculta se podría obtener un bajo error de entrenamiento, pero se tiene un alto error de generalización debido al overfitting (sobreajuste).

Otro tema a tener en cuenta es el número de capas ocultas, ya que al aumentar el número de capas ocultas, se puede disparar el número de parámetros. Además, la mayoría de los problemas se puede resolver de manera óptima en 1 o 2 capas ocultas, debido a que se ha demostrado que aunque el aprendizaje sea más rápido al principio, luego puede estabilizarse antes de encontrar el error mínimo.

Por otro lado, normalmente el número óptimo de neuronas en la capa oculta es de 2/3 la suma de neuronas de la capa de entrada y de salida.

## VII. ALGORITMO DE APRENDIZAJE

El algoritmo de Retropropagación se basa en el método de descenso del gradiente para acercarse a al mínimo error cuadrático.

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} \quad (6)$$

$$\Delta w_{ij} = \alpha \cdot \delta_j \cdot x_i \quad (7)$$

Donde $\delta_j$ es el error propagado, $x_i$ el valor de la entrada y $\alpha$ es el factor de aprendizaje.

Sin embargo, la función de error suele tener muchos mínimos locales, quedando el algoritmo atrapado en un mínimo local no deseado. Para prevenirlo, se puede hacer que los cambios de pesos sinápticos dependan del gradiente medio de los puntos de un entorno, en vez de depender de un solo punto. Pero dicha modificación suele requerir un gran esfuerzo computacional y no resultar eficiente. A continuación se muestran dos mejoras del algoritmo posibles.

### A. *Factor de aprendizaje adaptativo*

Esta modificación del algoritmo de aprendizaje, se basa en la modificación del factor de aprendizaje inicial,$\alpha$, adaptándose según las necesidades del aprendizaje de la siguiente forma:

- Si las pendientes tienen el mismo signo, los pesos cambiarán más rápido.

$$\alpha(t+1) = \alpha(t) + k \quad (8)$$

  Donde k se denomina valor de capa y se le asigna el valor de 0,035.

- Si las pendientes tienen distinto signo, los pesos cambiarán más lentamente.

$$\alpha(t+1) = \alpha(t) \cdot (1-\gamma) \quad (9)$$

  Donde $\gamma$ tiene el valor de 0,333.

### B. *Aprendizaje por momentos*

Rumelhart, Hinton y William (1986) [21] propusieron que se tuviera en cuenta el gradiente de la iteración anterior y realizar un promedio con el gradiente de la iteración actual. Dicho promedio produce una reducción drástica de las fluctuaciones del gradiente en iteraciones consecutivas evitando que el algoritmo sea tan lento y por tanto poco eficiente.

$$\Delta w_{ij} = \alpha \cdot \delta_j \cdot x_i + \mu \cdot \left(w_{ij}(t) - w_{ij}(t-1)\right) \quad (10)$$

Donde $\mu$ es el momento, que suele tener el valor de 0,9 como valor óptimo.

La inclusión del Momento en el algoritmo de Retropropagación tiende a acelerar la bajada en direcciones similares al descenso, mientras que si se tiene oscilaciones de signo en iteraciones consecutivas, se ajustará el peso en cantidades pequeñas, actuando como estabilizador.

## VIII. PRUEBAS

En el ámbito del proyecto RECLAMO, se ha comenzado la validación de las distintas partes de la arquitectura propuesta. En concreto, se han realizado diversas pruebas de concepto con redes pequeñas para la validación de la eficacia de clasificación obtenida por el algoritmo propuesto en este artículo. Todas ellas han sido satisfactorias, obteniendo clasificaciones correctas.

No obstante, las siguientes fases de validación se están realizando para garantizar la viabilidad del algoritmo con respecto a la clasificación de respuestas dado el contexto.

De manera aproximada, teniendo en cuenta lo mencionado en el apartado VI.E. , para la realización del entrenamiento con una capa oculta, ha sido necesario recoger como mínimo unas 200 muestras compuestas de diferentes contextos recogidos tras introducir en el sistema distintos tipos de ataque y respuestas posibles.

Concretamente se está realizando la batería de pruebas con los tipos de ataques que se han clasificado dentro de la ontología: Backdoors, Buffer Overflow, DoS, Exploits, Mapping, Scanning, Sniffing, Hijacking, Spoofing, Cookie Poisoning, Cross Site Scripting, Parameter Tampering, SQL Injection, Brute Force, Diccionary Attack, Trojan, Virus, Worms.

## IX. CONCLUSIONES

En este artículo se propone el uso de aprendizaje automático para entrenar al sistema de respuestas a intrusiones desarrollado. Concretamente, se ha optado por el uso del algoritmo de Retropropagación ya que al ser un algoritmo supervisado el entrenamiento de la red neuronal convergerá más rápidamente.

Por otro lado, el uso de esta tecnología nos permite la clasificación de cualquier respuesta, sea cual sea la intrusión que llego al sistema. Esto quiere decir que se pueden obtener buenos resultados aunque se presenten intrusiones desconocidas para el sistema; ya que este algoritmo se ha desarrollado de manera independiente al tipo de intrusión.

Una vez que tenemos entrenada la red, este algoritmo se puede implantar en cualquier host independientemente de cual sea su uso o importancia como activo de la organización.

Como trabajo futuro inmediato se está realizando la validación del uso de dicho algoritmo para la clasificación de respuestas del AIRS. Para ello, la implementación se ha realizado de manera totalmente parametrizable, de manera que podamos realizar estudios de rendimiento de distintas arquitecturas de red neuronal y así obtener los resultados más óptimos.

Además de esto, como trabajo futuro se propone el uso de otras técnicas de mejora del algoritmo como son:

- La inicialización de pesos aleatoria basada en rango
- Utilización del método SAB que combina momento y aprendizaje adaptativo.

Por último, se estudiarán las posibilidades para desarrollar respuestas proactivas en el sistema propuesto. Con esto, podremos adelantarnos a los pasos del atacante para evitar de manera anticipada posibles intrusiones.

## AGRADECIMIENTOS

Este trabajo ha sido financiado parcialmente con el apoyo del MICINN Español (Proyecto RECLAMO, Virtual and Collaborative Honeynets based on Trust Management and Autonomous Systems applied to Intrusion Management, con códigos TIN2011-28287-C02-01 y TIN2011-28287-C02-02).

## REFERENCIAS

[1] Symantec Corp., “Internet Security Threat Report, Vol. 17,” Abril 2012.

[2] Anderson, James. “Computer Security Threat Monitoring and Surveillance”. Washing, PA, James P. Anderson Co. 1980.

[3] H. J. Mattord. “Principles of Information Security Course Technology”. 2008. ISBN 9781423901778: 290-301.

[4] Ali Aydin M, Halim Zaim A, Gökhan Ceylan K. “A hybrid intrusion detection system design for computer network security”. Comput Elect Eng, 2009; 35(3):517–26.

[5] Stakhanova N, Basu S, Wong J. “A taxonomy of intrusion response system.” Int J Inform Comput Secur. 2007; 1(1/2):169–84

[6] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, “SWRL: A semantic web rule language combining OWL and RuleML”. W3C Member Submission, 21, 2004.

[7] J. E. López de Vergara, E. Vázquez, A. Martin, S. Dubus, M. N. Lepareux. “ Use of ontologies for the definition of alerts and policies in a network security platform”, Journal of Networks, Vol. 4, Issue 8 (2009) pp. 720-733

[8] H. Debar, D. Curry, B. Feinstein. “The Intrusion Detection Message Exchange Format (IDMEF)”. IETF Request for Comments 4765, Marzo 2007

[9] J.E. López de Vergara, V.A.Villagrá, J.I. Asensio, J. Berrocal. “Ontology-based network management: study cases and lessons learned”. J Network Syst Manage 2009; 17(3):234–54.

[10] D. L. McGuinness, F. van Harmelen. “OWL Web Ontology Language Overview. “ W3C Recommendation 10 Febrero 2004

[11] Hu, Xunlei Rose, and Eric Atwell. "A survey of machine learning approaches to analysis of large corpora." Proceedings of the Workshop on Shallow Processing of Large Corpora, Lancaster University, UK. 2003.

[12] V. Mateos, V.A. Villagrá, F. Romero. “Ontologies-Based Automated Intrusion Response System. “ Computacional Intelligence in Security for information Systems 2010. Volume 85/2010. 2010:99/106

[13] V. Mateos. V. A. Villagrá, F. Romero, J. Berrocal. “Definition of response metrics for an ontology-based Automated Intrusion Response Systems.” Computers & Electrical Engineering. 2012

[14] J. Heaton. “Introduction to Neural Networks for Java”. 2nd Edition.

[15] Nagios Core Tool : http://www.nagios.org/projects/nagioscore

[16] Ling-xi Peng, Dong-qing Xie, Ying Gao, Wen-bin Chen, Fu-fang Li, Wu Wen. “An Inmune-inspired Adaptative Automated Intrusion Response System. International Journal of Computational Intelligence” Systems, 2012, 5:5, 808-815.

[17] Alireza Shameli-Sendi, Julien Desfossez, Michel Dagenais, Masoume Jabbarifar. “A Retroactive- Burst Framework for Automated Intrusion Response System”, Journal of Computer Networks and Communications, Volume 2013.

[18] Justina, Aderonke, and Adesina Simon. “A credible cost-sensitive model for intrusion response selection.” In Computational Aspects of Social Networks (CASoN), 2012 Fourth International Conference on, pp. 222-227. IEEE, 2012.

[19] Z. Mahmood, C. Agrawal, S. S. Hasan, S. Zenab, “Intrusion Detection in Cloud Computing environment using Neural Network”. International Journal of Research in Computer Engineering & Electronics, 2012. 1(1), 19-22.

[20] R. S. Naoum, Abid, N. A., Z. N. Al-Sultani, “An Enhanced Resilient Backpropagation Artificial Neural Network for Intrusion Detection System”. IJCSNS, 2012. 12(3), 11.

[21] Rumelhart, D.E., Hinton, G.E. y Williams, R.J. (1986). Learning internal representations by error propagation. En: D.E. Rumelhart y J.L. McClelland (Eds.). Parallel distributed processing (pp. 318-362). Cambridge, MA: MIT Press