UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

Sistema de Respuesta a Intrusiones Proactivo basado en modelos de Markov y Redes Neuronales

TESIS DOCTORAL

Pilar Holgado Ortiz

Máster en Ingeniería Informática y de Telecomunicación

2018

Departamento de Ingeniería de Sistemas Telemáticos

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE MADRID

Sistema de Respuesta a Intrusiones Proactivo basado en modelos de Markov y Redes Neuronales

TESIS DOCTORAL

Autor

Pilar Holgado Ortiz

Máster en Ingeniería Informática y de Telecomunicación

Director

Víctor A. Villagrá Gonzalez

Doctor Ingeniero de Telecomunicación

Madrid, Enero 2018

Título: Sistema de respuesta a intrusiones proactivo basado en modelos de Markov y Redes Neuronales

Departamento: Departamento de ingeniería de sistemas telemáticos

Autor: Pilar Holgado Ortiz

Director: Víctor A. Villagrá González

TRIBUNAL CALIFICADOR

Presidente:	D. Julio Berrocal Colmenarejo	Doctor Ingeniero de Telecomunicación Catedrático de Universidad Universidad Politécnica de Madrid
Vocales:	D. Jorge E. López de Vergara Méndez	Doctor Ingeniero de Telecomunicación Profesor Titular Universidad Universidad Autónoma de Madrid
	Dña. Lorena Gonzalez Manzano	Doctor en Ciencia y Tecnología Informática Profesor Visitante Universidad Carlos III de Madrid
	D. Gregorio Martinez Perez	Doctor Ingeniero en Informática Catedrático de Universidad Universidad de Murcia
Secretario:	D. Gregorio López López	Doctor en Ingeniería Telemática Profesor Ayudante Doctor Universidad Politécnica de Madrid
Suplentes:	D. Luis Javier García Villalba	Doctor Ingeniero en Informática Profesor Contratado Doctor Universidad Complutense de Madrid
	Dña. Marta Beltran Pardo	Doctor Ingeniero en Informática Profesor Titular Universidad Universidad Rey Juan Carlos de Madrid

Sigue nadando...

Agradecimientos

En primer lugar me gustaría dar las gracias a mi director de tesis Víctor Villagrá y al grupo de investigación RSTI por hacer posible la realización de esta tesis doctoral.

También quiero resaltar el apoyo de mi familia y amigos durante este periodo. Destacando a mis padres por darme fuerzas en este duro camino. Y finalmente, a mi marido por estar a mi lado en todo momento y levantarme el ánimo para seguir adelante a pesar de los escollos que fueron surgiendo.

Sistema de Respuesta a Intrusiones Proactivo basado en modelos de Markov y Redes Neuronales

Resumen

Los incidentes de seguridad han aumentado en gran medida debido al uso extendido de tecnologías de la información (TI) dentro de todos los ámbitos de las organizaciones. Además, estos incidentes cada vez son más sofisticados por lo que los sistemas de protección deben evolucionar para detectar y mitigar estos incidentes. Los Sistemas de Detección de Intrusiones han evolucionado rápidamente y a día de hoy existen diversidad de herramientas maduras basadas en distintos paradigmas. A su vez, los Sistemas de Prevención de Intrusiones se han incluido para la realización de respuestas reactivas básicas, como restablecer una conexión. Por otro lado, se han ido desarrollando Sistemas de Respuestas a Intrusiones (IRS) con el objetivo de proporcionar respuestas específicas de acuerdo a unas reglas predefinidas.

Hoy en día los IRSs juegan un rol importante en la arquitectura de seguridad. Estos sistemas mitigan el impacto de posibles ataques con el objetivo de mantener la integridad, la disponibilidad y la confidencialidad de los recursos. Los Sistemas de Respuestas a Intrusiones Automáticos (AIRS) proporcionan la mejor defensa posible mediante la selección automática de respuestas, además de minimizar la ventana de oportunidad del atacante. Esta tesis doctoral se centra en mejorar las reacciones automatizadas contra intrusiones usando métodos de Aprendizaje Automático.

La primera contribución de la tesis consiste en evaluar la eficiencia de las respuestas eje-

cutadas previamente frente a un ataque. De este modo, es posible proporcionar información relevante para la inferencia de siguientes respuestas. Para determinar la efectividad de cada una de las respuestas se ha optado por el uso de un algoritmo de aprendizaje basado en Redes Neuronales Artificiales, de tal forma que el AIRS tenga capacidad de auto-aprendizaje. Posteriormente, se define una serie de ecuaciones que determinan el nivel de éxito de la respuesta todas las veces que fue ejecutada. El valor del nivel de éxito de cada una de las respuestas es almacenado en la Ontología del AIRS para así poderlo tener en cuenta en los siguientes procesos de inferencia. Como consecuencia se consigue un AIRS con capacidades adaptativas.

El concepto de predicción de ataques es clave para adelantarse a los pasos del atacante, y así poder realizar respuestas proactivas. Por tanto, la segunda propuesta de la tesis doctoral se basa en predecir ataques multi-paso a partir de las alertas reportadas por los Sistemas de Detección de Intrusiones (IDS) diseñando un sistema basado en la definición de un modelo extendido de los Modelos de Markov Ocultos (HMM). Los estados ocultos de la cadena de Markov son las fases de ataque generalizadas para cada uno de los distintos tipos de ataques a modelar. De esta manera, el modelo se adapta al ataque multi-paso concreto y como resultado es posible adelantarse al siguiente paso del atacante.

Específicamente, la validación del sistema predictivo propuesto se va a centrar en el análisis de las distintas fases de la Denegación de Servicio Distribuida (DDoS) ya que es un problema creciente en los servicios de Internet y es complicado prevenir caídas de los sistemas. Para conseguir este objetivo es necesario que el HMM sea entrenado de manera off-line a partir de una serie de observaciones. Estas observaciones se obtienen tras el etiquetado de

los distintos identificadores de CVE (Common Vulnerabilities and Exposures) que puede contener una alerta para así evitar problemas de sobreajuste. El etiquetado se basa en la clusterización del informe público de CVE, los cuales son posteriormente almacenados en una base de datos.

Por otro lado se compararán dos algoritmos distintos de entrenamiento, supervisado y no supervisado para la construcción de las matrices de probabilidad. Tras ello el modelo está preparado para recibir alertas de distintos IDSs y encontrar la mejor secuencia de estados usando el algoritmo de Viterbi. La probabilidad de que el ataque se encuentre en un estado concreto para cada uno de los distintos pasos del ataque multi-paso en progreso está basado en el algoritmo de Viterbi y en el algoritmo de forward-backward. Finalmente se calcula la probabilidad de que se produzca el objetivo final del ataque usando el número medio de alertas obtenidas en el entrenamiento y el número de alertas en progreso. El sistema propuesto se ha validado con un escenario virtual construido teniendo en cuenta vulnerabilidades actuales y se ha llevado a cabo medidas del rendimiento del sistema predictivo, verificando la viabilidad de obtener valores de predicción en tiempo real.

Palabras clave Red Neuronal Artificial, algoritmo de Retropropagación, predicción de ataques multi-paso, Modelo de Markov Oculto, Denegación de Servicio Distribuido, respuesta proactiva, aprendizaje automático.

Sistema de Respuesta a Intrusiones Proactivo basado en modelos de Markov y Redes Neuronales

Abstract

Security incidents have increased greatly due to the widespread use of information technology (IT) within any organizational environment. As the number of security incidents increases, becoming more sophisticated and widespread, Intrusion Detection Systems (IDS) have evolved rapidly and there are now very mature tools based on different paradigms. Intrusion Prevention Systems have also been developed by combining IDS with a basic reactive response, such as resetting a connection. IRSs (Intrusion Response Systems) leverage the concept of IPSs and provide the means to achieve specific responses according to some predefined rules.

Nowadays, Intrusion Response Systems are playing an important role in the security architecture. These systems mitigate the impact of attacks in order to keep integrity, confidentiality and availability of the resources. Automated Intrusion Response Systems (AIRS) provide the best possible defense, as well as shortening the delay before administrators come into play. This dissertation improvement the automated reactions against intrusions using machine learning methods.

The first propose is evaluating the result of previous responses, in order to get feedback for following inferences. This thesis defines an algorithm to determine the level of success of the inferred response. The objective is the design of a system with adaptive and self-learning capabilities. Artificial Neural Networks are able to provide machine learning in order to get

responses classification.

Attack prediction is an important method of foreseeing the steps of an attacker and thus, AIRS can execute proactive responses. A novel method based on a extension of the Hidden Markov Model (HMM) definition is design as a second propose. This proposal predicts multi-step attacks using intrusion detection system (IDS) alerts. We consider the hidden states as common phases of a particular type of attack. As a result, it can be easily adapted to multi-step attacks and foresee the next steps of that particular attack.

Specifically, we analyze the phases of DDoS (Distributed Denial of Service). DDoS is a great problem in all Internet services. Currently, there is no way to prevent a possible server crash. To achieve this goal, a preliminary off-line training phase based on observations will be required. These observations are obtained by matching the IDS alert information with a database previously built for this purpose using a clusterization method from the Common Vulnerabilities and Exposures (CVE) global database to avoid overfitting. The training model is performed using both unsupervised and supervised algorithms.

Once the training is completed and probability matrices are computed, actual IDS alerts will trigger the prediction module by finding the best state sequence using the Viterbi algorithm. The state probability for each step of the multi-step attack in progress is based on the Viterbi and forward-backward algorithms. The training model includes the mean number of alerts and the number of alerts in progress to assist in obtaining the final intrusion probability. The proposed method is validated into a virtual DDoS scenario using current vulnerabilities and we probe the system's ability to perform real-time prediction.

Keywords Artificial Neural Network, Backpropagation algorithm, multi-step attack prediction, Hidden Markov Model, Distributed Denial of Service, proactive response, machine learning

Índice

Índice de figuras

Índice de tablas

1 INTRODUCCIÓN

1.1 Introducción y motivación

Los ciberataques son una amenaza sobre todos los sistemas que componen una organización. Cada vez más, las organizaciones están concienciadas en invertir en sistemas que incrementen la seguridad, 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.

Los Sistemas de Detección de Intrusiones (IDS), han evolucionado rápidamente, y en la actualidad, hay herramientas muy sofisticadas basadas en los distintos paradigmas (estadísticas basadas en anomalías, basadas en firmas e híbridas) 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) [SSEJJD12] aprovechan el concepto de IPS para lograr una respuesta específica de acuerdo a unas reglas predefinidas. Un Sistema de Respuestas

Automático (AIRS) proporciona la mejor defensa posible y acorta la ventana de oportunidades hasta que el administrador del sistema puede tomar un rol activo en defender contra el ataque.

Uno de los inconvenientes de la automatización de los sistemas actuales es que tienen un enfoque fijo para las métricas de respuesta, por lo que las respuestas no pueden ser elegidas de manera dinámica ni aprender de dichas respuestas para eventos posteriores. En esta tesis se propone mejorar la arquitectura de seguridad propuesta en [ML13], 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.

Por otro lado, a pesar de estos y otros esfuerzos por proteger los sistemas, se ha incrementado el número de eventos de seguridad y su sofisticación [CC16]. Uno de los ataques mas extendidos hoy en día es la Denegación de Servicio Distribuida (DDoS). Actualmente hay varios ejemplos de compañías que han sufrido importantes ataques de denegación de servicio como los dirigidos contra las plataformas de juegos online tales como *Pokemon Go*, *League of Legends* y *battle.net* , o contra compañías de alojamiento en todo el mundo como la compañía francesa de hosting *OVH* [Cer14].

La Denegación de Servicio Distribuida es un claro ejemplo de ataque multi-paso [Ken99], esto es, un escenario de ataque que lleva a cabo el mismo atacante y que se compone de varias actividades maliciosas para conseguir un objetivo final. La identificación de estos pasos puede utilizarse para predecir un ataque de manera temprana.

Los sistemas actuales no son capaces de adelantarse a los pasos del atacante antes de que consiga su objetivo final. De esta limitación surge la siguiente propuesta de tesis, que tiene como objetivo conseguir un sistema de predicción de intrusiones.

Este modelo predictivo puede ser utilizado para mejorar tanto los Sistemas de Respuestas a Intrusiones, como los Sistemas de Gestión del Riesgo Dinámico. El Proceso de Gestión de Riesgos se encarga de evaluar los riesgos de los activos que componen la red organizativa y de la toma de decisiones para hacer frente a dicho riesgo. Puesto que el riesgo es un concepto dinámico que varía con el transcurso del tiempo y con las modificaciones en los múltiples

factores que definen dicho riesgo, la Gestión del Riesgo establece el seguimiento y el análisis continuo de su evolución.

1.2 Objetivos de la tesis doctoral

En esta tesis doctoral se propone mejorar la arquitectura de seguridad propuesta en [ML13]. La tesis puede dividirse en dos objetivos principales relacionados con la respuesta a intrusiones: el autoaprendizaje basado en la efectividad de la respuesta y la predicción de futuros pasos de un ataque. Ambas propuestas de tesis están ligadas al Aprendizaje Automático.

1.2.1 Automatización y aprendizaje para respuestas a intrusiones

Muchos de los sistemas actuales de respuestas a intrusiones automáticos carecen de mecanismos de autoaprendizaje para la modificación dinámica de métricas de seguridad, sin tener en cuenta respuestas anteriormente lanzadas y que contribuyen a obtener respuestas más acertadas y rápidas frente a siguientes intrusiones del mismo tipo.

Por tanto, uno de los objetivos de la tesis consistirá en proporcionar capacidad de aprendizaje a los Sistemas de Respuestas a Intrusiones Automáticos (AIRS), siendo así posible la evaluación de las respuestas lanzadas previamente para su uso la siguiente vez que llegue una intrusión del mismo tipo.

1.2.2 Sistemas de predicción de intrusiones

Por otro lado, los sistemas actuales no son capaces de adelantarse a los pasos del atacante antes de que consiga su objetivo final. De esta limitación surge la siguiente propuesta de tesis, que tiene como objetivo conseguir un sistema de predicción de Intrusiones.

La capacidad de adelantarse a los acontecimientos de las distintas fases de intrusión posibilita la mitigación del ataque antes de que el sistema se vea totalmente afectado y que los activos de la organización se mantengan a salvo de las consecuencias de dichas acciones.

En la proactividad juega un papel principal el concepto de predicción de intrusiones. La predicción de intrusiones es un área de investigación imprescindible para poder afrontar

ciberataques en tiempo real y anticiparse a los posibles efectos del ataque sobre los activos de la organización. Por esta razón puede ser útil tanto en sistemas de respuesta para la respuesta temprana ante intrusiones o para la gestión del riesgo dinámico de una organización.

1.3 Metodología utilizada para el desarrollo de los objetivos de la tesis

Ambos objetivos de la tesis doctoral tienen en común la aplicación de un método de Aprendizaje Automático. El Aprendizaje Automático es una rama de la Inteligencia Artificial (IA) que ha sido utilizado en investigación en muchos ámbitos de la seguridad. Este tipo de métodos se basan en la aplicación de algoritmos que son capaces de aprender en base a experiencias previas como muestra de entrada. Dentro del gran abanico de métodos de Aprendizaje Automático existentes, tras el análisis de cada una de sus características y del estado del arte se determina que:

- *Objetivo 1:* Con el propósito del cálculo de la eficiencia de la respuesta se utilizará un algoritmo de aprendizaje automático bio-inspirado, concretamente un algoritmo basado en Redes Neuronales Artificiales (*Artificial Neural Network*, ANN).
- *Objetivo 2:* La predicción de ataques multi-paso se realizará basándose en aprendizaje automático de tipo estadístico, en concreto con Modelos de Markov Ocultos (*Hidden Markov Model*, HMM).

Las fases en las que se basa el desarrollo de la tesis para ambas contribuciones son las siguientes:

1. Partes comunes a ambos objetivos de la tesis doctoral (*Objetivo 1 y 2*):
   (a) Análisis del estado del arte de sistemas de detección y respuesta a intrusiones.
   (b) Análisis del estado del arte de mecanismos de Aprendizaje Automático.
   (c) Arquitectura global de la propuesta de la tesis doctoral.
2. Automatización y aprendizaje de respuestas a intrusiones. (*Objetivo 1*):

(a) Análisis del estado del arte de los algoritmos basados en Redes Neuronales

(b) Implementación del Algoritmo de Evaluación

(c) Validación del Algoritmo de Evaluación

(d) Integración en el Sistema de Respuestas a Intrusiones Automático desarrollado en el proyecto RECLAMO.

(e) Entrenamiento de la Red Neuronal.

(f) Validación del funcionamiento completo del sistema.

3. Sistema de predicción de intrusiones. (*Objetivo 2*):

(a) Estado del arte de Sistemas de Respuestas Proactivas.

(b) Análisis del estado del arte de alertas multi-paso.

(c) Estado del arte en sistemas de predicción de intrusiones.

(d) Estado del arte del modelado de ataques multi-paso.

(e) Estado del arte de Modelos de Markov.

(f) Definición de la arquitectura del Sistema Proactivo/predictivo.

(g) Definición del Modelo de Markov Oculto.

(h) Implementación del módulo de predicción de intrusiones.

(i) Análisis de fases de distintos escenarios de ataque de Denegación de servicio distribuido (DDoS).

(j) Clusterización de vulnerabilidades basadas en el reporte de CVE (Common Vulnerabilities and Exposures).

(k) Creación de la base de datos de vulnerabilidades.

(l) Entrenamiento off-line de los escenarios de ataque.

(m) Validación de la predicción de intrusiones.

(n) Cálculo de la probabilidad final de intrusión teniendo en cuenta la fase de ataque a partir del HMM.

(o) Validación del sistema completo.

Esta metodología de trabajo se ha llevado a cabo en su totalidad, excepto el punto (e) y (f) del *Objetivo 1*, los cuales han sido estructurados para llevarlos a cabo como trabajo futuro. En la tesis se recoge los puntos llevados a cabo junto a la planificación de las fases que componen dichas partes de la metodología de desarrollo.

1.4 Estructura de la memoria

A continuación se indica la estructura de la memoria en capítulos, teniendo en cuenta los puntos de la metodología en los que se desarrolla durante el trabajo de la tesis doctoral:

- *Capítulo 1:* Este capítulo introduce la motivación y los objetivos que se van a abordar en la tesis doctoral.
- *Capítulo 2:* Este capítulo trata sobre el estado del arte en ciberseguridad, incluyendo los ataques informáticos y su tipología; las características de un ataque multi-paso; y los distintos sistemas de protección contra intrusiones, sus características, diferencias y deficiencias. Este capítulo se ha llevado a cabo en los puntos de la metodología 1(a), 3(a), 3(b), 3(c) y 3(d).
- *Capítulo 3:* En este capítulo se especifican los mecanismos de aprendizaje automático centrándonos en las Redes Neuronales y los Modelos de Markov Ocultos que son las técnicas de aprendizaje que se utilizarán para el desarrollo de la tesis doctoral. Este estudio se realiza en los puntos 1(b), 2(a) y 3(e) de la metodología de trabajo.
- *Capítulo 4:* Se muestra la arquitectura global del sistema desarrollado para la tesis doctoral. De este modo se facilita la visualización de las dos contribuciones de tesis y sus relaciones dentro de un entorno de seguridad. Diseñado en el punto 1(c) de la metodología.
- *Capítulo 5:* Incluye la primera contribución de la tesis doctoral que consiste en la automatización y el aprendizaje sobre respuestas a intrusiones. Se detalla la implemen-

tación de la Red Neuronal Artificial, los métodos utilizados para la evaluación de respuestas y el entrenamiento. Finalmente, se planifica las fases del entrenamiento y la validación a realizar. Todo este desarrollo se ha realizado en el punto 2 de la metodología, en concreto en las partes (b), (c) y (d)

- *Capítulo 6:* Detalla la segunda contribución de la tesis, el sistema de predicción de intrusiones. En concreto, se muestra el diseño de la arquitectura necesaria para la predicción de ataques multipaso y se define el modelo a utilizar, junto con el cálculo de probabilidades. Por otro lado, el sistema predictivo ha sido validado en varios entornos de DDoS y se han realizado pruebas con distintos métodos de entrenamiento. Este capítulo se centra en el punto 3 de la metodología de desarrollo, en concreto todas las partes incluidas entre la (f) hasta la (o).

- *Capítulo 7:* Contiene el análisis de las conclusiones de las contribuciones de tesis y la línea de trabajo futura.

2

CIBERSEGURIDAD

2.1 Introducción

La seguridad en cualquier organización es imprescindible para mantenerse a salvo de ataques y mantener sus niveles de servicio. La seguridad informática se puede entender como que los sistemas de una organización están libres de peligro; aunque esto no es así debido a que es utópico pensar que es posible que un sistema esté 100 % seguro, puesto que día a día surgen nuevas amenazas más sofisticadas [Cor17] para los cuales, los mecanismos de seguridad no están aún preparados y por tanto deben estar en constante evolución.

Dentro de la seguridad informática es necesario tener claros una serie de conceptos:

1. Activos: Todos los recursos que posee una organización y que son susceptibles de ser atacados, como por ejemplo un servidor web.

2. Amenaza: Evento que puede desencadenar un incidente en la organización, produciendo daños en sus activos.

3. Vulnerabilidad: Posibilidad de que una amenaza se materialice sobre un activo.
4. Impacto: Consecuencia de la materialización de la amenaza sobre un activo.
5. Riesgo: Posibilidad de que se produzca un impacto en un activo.
6. Salvaguarda: Medidas destinadas a mitigar la importancia de los riesgos. Hay distintos tipos, preventivas, defensivas y proactivas.
7. Ataque: Evento que intenta atentar sobre el funcionamiento de un activo.

En este capítulo se analizan distintos tipos de ataques, las características de los sistemas de protección y el estudio del estado del arte sobre sistemas de defensa; estructurándose en cuatro partes bien diferenciadas:

- Fundamentos en ciberseguridad dentro del entono de la Tecnología de la Información.
- Análisis de distintos tipos de amenazas y ataques.
- Características de algunos sistemas de defensa perimetral.
- Estudio de distintas propuestas previas en el entorno de investigación de los sistemas de defensa perimetral.

2.2 Servicios de Seguridad

La seguridad informática es un conjunto de actuaciones que permiten proteger un entorno, asegurando de esta manera los recursos del sistema de información de una institución.

Para que un sistema pueda considerarse seguro debe de cumplir una serie de características [Sta04]:

- Integridad: La información solo puede ser modificada por las personas autorizadas.
- Confidencialidad: La información sólo debe ser legible por los usuarios autorizados.
- Disponibilidad: La información o el servicio debe estar siempre accesible para los usuarios autorizados.

- Autenticidad: El emisor de la información debe ser quién escribió el mensaje y no debe ser posible negar la autoría del mismo.

Existen diversas maneras de combatir ataques en los activos de la organización, éstas pueden abstraerse en una serie de niveles de actuación:

1. *Nivel de prevención*: Se trata de todas aquellas medidas de seguridad que están destinadas a evitar que un ataque o una amenaza se materialice. En general, en este nivel se encuentran todas las medidas de defensa perimetral: cortafuegos, filtros de aplicación (proxies o wrappers) o medidas de control de acceso.

2. *Nivel de detección*: Se trata de todos aquellos medios que permiten conocer que, efectivamente, se ha producido una violación de las medidas de seguridad de los recursos que estaban siendo protegidos. En principio podría parecer que no aportan nada a la seguridad de estos recursos, ya que si las medidas del nivel de prevención no han conseguido evitar dicho ataque, éstas, evidentemente, tampoco podrán hacerlo. Sin embargo, este tipo de medidas que permiten conocer el estado de los recursos y sus posibles modificaciones son de especial utilidad como complemento (no como sustitutas) de las anteriores.

3. *Nivel de reacción*: Un tercer nivel lo constituyen los mecanismos que permiten denunciar o dar a conocer que un ataque se ha materializado sobre un determinado recurso o incluso tomar las medidas adecuadas contra el origen de dicho ataque. En todo caso, puede apreciarse claramente que estas medidas están íntimamente relacionadas con las anteriores, ya que la reacción sólo es posible tras la detección.

Estos son los niveles clásicos aplicados por los distintos sistemas de protección, en esta tesis se propone añadir un nivel de actuación, el *nivel de predicción*, que se encargue de detectar de manera anticipada los posibles pasos que podría dar el atacante para que los sistemas de protección de la organización puedan reaccionar de manera proactiva y adelantarse a los posibles efectos negativos en los activos de la organización.

Se pueden distinguir dos tipos de sistemas de protección contra accesos malintencionados [VGML09]:

- Sistemas de autenticación y autorización: Permiten el control de acceso lógico a los servicios proporcionados por la organización a los usuarios autorizados, tanto externos como internos. Estos sistemas se utilizan para la restricción del acceso a los activos de la organización y así evitar en lo posible ataques de usuarios no autorizados o que no tenga nivel de acceso suficiente para acceder a dicho activo o información.
- Sistemas de defensa perimetral: Restringir accesos externos no permitidos a los servicios externos de una organización protegida. Estos sistemas se encargan de la prevención, la detección y la respuesta frente a ciberataques. En el caso de hogares esta solución suele centrarse en el uso de un pequeño *firewall*, pero para organizaciones estos sistemas pueden llegar a ser mucho más complejos, actuando en cada uno de los sistemas o como entrada a cada una de las subredes de la organización.

Esta tesis doctoral se enfoca en los sistemas de defensa perimetral. Para restringir el paso del tráfico no deseado estos sistemas utilizan distintos mecanismos de protección, como pueden ser, cortafuegos, sistemas de detección de intrusiones (*Intrusion Detection System*, IDS), *honeypots* e inspectores de contenido.

2.3 Características de las amenazas y ciberataques

Las inseguridades a las que están expuestos los usuarios y sistemas en Internet se pueden clasificar de la siguiente manera:

1. Amenazas por ser alcanzables en internet:
   (a) Acceso remoto no autorizado:
      i. Aprovecha una vulnerabilidad del software: Frecuentemente utilizados para ataques de desbordamiento de buffer.
      ii. Aprovecha vulnerabilidades de configuración del servidor de red

iii. Aprovecha una vulnerabilidad en la implementación de la torre de protocolos de acceso al servicio

(b) Ataques de negación de servicios inundando a peticiones al servidor (DoS) violando la disponibilidad del servicio a los usuarios autorizados.

2. Amenazas por actividad de un usuario en internet

(a) Las vulnerabilidades de las aplicaciones pueden ser aprovechadas por los atacantes para, por ejemplo, el acceso a información privada.

(b) Ataques de ingeniería social: Consisten en engañar al usuario para que realice acciones que permitan la realización de un ataque. Para ello los atacantes pueden utilizar técnicas como la suplantación de páginas web (*web spoofing*), la suplantación de identidades (*phishing*), o incluso proporcionar a los usuarios medios de almacenamiento, como una memoria USB, con programas maliciosos, como por ejemplo un programa espía (*spyware*).

(c) Acceso de los atacantes a la información en tránsito. En este caso se pueden diferenciar distintos tipos de ataques: de disponibilidad, confidencialidad, integridad, autenticidad y repudio.

Un ataque informático es cualquier actividad maliciosa dirigida a sistemas de información o a los servicios que ellos proporcionan. Ejemplos de ataques son, el uso de un virus para usar un sistema sin autorización, la denegación de servicio explotando un bug, escaneo de un sistema para obtener información o un ataque físico contra el hardware de un ordenador.

Los ataques se pueden clasificar dependiendo de los siguientes factores:

1. Experiencia del atacante

(a) Ataques estructurados: Este tipo de ataque son realizados por atacantes técnicamente competentes y con una gran motivación para conseguir el objetivo. Tienen conocimiento de las vulnerabilidades de un sistema, por lo que desarrollan códigos o scripts para explotar dichas vulnerabilidades. Utilizan técnicas de hacking

sofisticadas, por ejemplo para penetrar en empresas sin ser detectado. Muchas veces actúan en grupos que por lo general están involucrados en grandes fraudes.

(b) Ataques no estructurados: Estos ataques son llevados a cabo por individuos sin experiencia que utilizan herramientas de hacking fáciles de encontrar, como por ejemplo scripts de *shell* y rompedores de clave. Muchas veces se realizan para poner a prueba habilidades, pero pueden repercutir en daños graves.

2. Origen del ataque

   (a) Ataques externos: Son realizados por uno o varios atacantes desde fuera de una empresa. En este sentido, ninguno de ellos tiene acceso legítimo al sistema o a la red que están intentando atacar. Para acceder desde el exterior muchas veces lo realizan a través de servidores accesibles desde internet.

   (b) Ataques internos: Son los ataques más peligrosos ya que son realizados por individuos con acceso autorizado a la red o sistema. Normalmente se asocian a empleados descontentos.

3. Interacción sobre el flujo de comunicación

   (a) Ataques pasivos: El atacante no altera la comunicación, sino que únicamente escucha o monitoriza la red para obtener información que está siendo transmitida. Los ataques pasivos son muy difíciles de detectar, ya que no provocan ninguna alteración de los datos. Sin embargo, es posible evitar su éxito mediante el cifrado de la información. Sus objetivos son la intercepción de datos y el análisis de tráfico, una técnica más sutil para obtener información de la comunicación, que puede consistir en:

      i. Obtención del origen y destinatario de la comunicación, leyendo las cabeceras de los paquetes monitorizados.

      ii. Control del volumen de tráfico intercambiado entre las entidades monitorizadas, obteniendo así información acerca de actividad o inactividad inusuales.

iii. Control de las horas habituales de intercambio de datos entre las entidades de la comunicación, para extraer información acerca de los períodos de actividad.

(b) Ataques activos: Estos ataques implican algún tipo de modificación del flujo de datos transmitido o la creación de un falso flujo de datos, pudiendo subdividirse en cuatro categorías:

i. Suplantación de identidad: el intruso se hace pasar por una entidad diferente. Normalmente incluye alguna de las otras formas de ataque activo. Por ejemplo, las secuencias de autenticación pueden ser capturadas y repetidas, permitiendo a una entidad no autorizada acceder a una serie de recursos privilegiados suplantando a la entidad que posee esos privilegios, como al robar la contraseña de acceso a una cuenta.

ii. Reactuación: uno o varios mensajes legítimos son capturados y repetidos para producir un efecto no deseado, como por ejemplo ingresar dinero repetidas veces en una cuenta dada.

iii. Modificación de mensajes: una porción del mensaje legítimo es alterado, o los mensajes son reordenados. Por ejemplo, el mensaje *Ingresa un millón de euros en la cuenta A* podría ser modificado para decir *Ingresa un millón de euros en la cuenta B*.

iv. Degradación fraudulenta del servicio: impide o inhibe el uso normal o la gestión de recursos informáticos y de comunicaciones. Por ejemplo, el intruso podría suprimir todos los mensajes dirigidos a una determinada entidad o se podría interrumpir el servicio de una red inundándola con gran cantidad de peticiones. Entre estos ataques se encuentran los de denegación de servicio, consistentes en paralizar temporalmente el servicio de un servidor de correo, Web, FTP, etc.

2.4 Ataques multi-paso

Un escenario de ataque o ataque multi-paso son un conjunto de actividades maliciosas relacionadas y ejecutadas para conseguir un objetivo específico. La relación entre dichas actividades maliciosas es una relación causal, es decir, una actividad maliciosa es precondición de la siguiente. Hay diversos tipos de escenarios de ataque [ZGHS09] como el escalado remoto de privilegios, gusanos, botnet, *phishing*, Denegación de Servicio Distribuida (*Distributed Denial of Service*, DDoS).

A continuación se va ha detallar el proceso de las fases que realiza el atacante para algunos ejemplos de ataques multi-paso:

1. WannaCry: En Mayo de 2017 se detectó un nuevo *ransomware* conocido como WannaCry [Mic17] que se expande por la red como un gusano aprovechando vulnerabilidades conocidas. Específicamente, la amenaza llega como un troyano con tres fases:

   (a) Buscar máquinas vulnerables a SMB EternalBlue.

   (b) Explotar la vulnerabilidad SMB EternalBlue (*CVE 2017-0145*).

   (c) Instalar el *ransomware* WannaCry.

2. *Local to Root* (L2R): El ejemplo se basa en la penetración de una máquina Linux para conseguir privilegios de root. El atacante explota un fallo en el correo electrónico, en el que no se reestablece el bit *setuid* del fichero en el cual se añade un mensaje y cambia el propietario del mismo. Como resultado el atacante es capaz de engañar al correo y crear un programa *shell* setuid que es propiedad de root y ejecutable públicamente [YF07].

   (a) El atacante crea una copia de *csh* en el archivo de mail de root. Para que esta etapa sea exitosa, el atacante debería esperar hasta que root no tenga correo no leído, de lo contrario el atacante no será capaz de crear el archivo de correo falsificado. (`cp /bin/csh /usr/spool/mail/root`).

(b) El atacante permite que el bit *setuid* del archivo de correo sea falsificado. (chmod 4755 /usr/spool/mail/root).

(c) El atacante crea y envía un mensaje vacío a root vía la herramienta mail. (touch x y mail root <x).

(d) El atacante solo necesita ejecutar el mail de root para ganar acceso a una *shell* con privilegios de root. (/usr/spool/mail/root).

3. *Remote to Local* (R2L):

(a) [SSD15] propone un ataque de *R2L* donde el atacante escanéa la red (*probes the network*), accede a la base de datos explotando la vulnerabilidad de un servicio web de otro servidor (*breaks into*) y establece una terminal (*reverse* shell) en su máquina local desde el host comprometido. Por tanto divide el escenario de ataque en cuatro fases:

i. Búsqueda de máquinas activas: El atacante envía *ICMP echo-requests* a la red y escucha los *ICMP echo-replies* para determinar los host que están activos.

ii. Escaneo de puertos y búsqueda de servicios disponibles: El atacante hace un escaneo de puertos y envía peticiones a un rango de puertos del servidor para ver cuales están abiertos. Tras esto, se analizan los puertos abiertos con servicios vulnerables para explotarlos.

iii. Aplica fuerza bruta para obtener usuarios y contraseñas: Primero, el atacante encuentra el fichero de configuración del sitio web que utiliza el servidor de base de datos. En el fichero de configuración se puede encontrar la *IP* del servidor de base de datos *MySQL* y un usuario y contraseña con permisos de solo lectura. Posteriormente tiene que conseguir acceso completo al servidor. El atacante encuentra la página de administrador con la contraseña segura utilizando la herramienta *skipFish*. Crea una puerta trasera (*backdoor*) al servidor web *Apache* para ejecutar comandos *shell*.

iv. Establecimiento del *reverse* shell: Una vez que el atacante ha comprometido la base de datos, crea una *shell inversa* para redirigir la salida de los comandos que ejecuta a su máquina a través del *pipe*.

(b) El mismo autor en [SSD14] y [Sen13] plantea otro escenario similar de *R2L* obteniendo privilegios de *root* (*U2R, user to root*):

i. Búsqueda de máquinas activas (*Probing*): El atacante analiza la red y escanea los puertos para encontrar servicios disponibles mediante la herramienta *nmap*. De esta manera, se encuentra un servidor web, así que el atacante utiliza la herramienta *Skipfish* para detectar fallos de seguridad. Como resultado, se encuentra que el foro *phpBB2* está disponible en el servidor.

ii. Explotar *phpBB2*: La versión *phpBB2 2.0.10* tiene la vulnerabilidad de inyección de *scripts* remoto, permitiendo al atacante ejecutar código *PHP arbitrario*. Como consecuencia, el atacante puede ejecutar cualquier comando sobre el servidor directamente como puede hacerlo un usuario *Apache* (*CVE 2005-2086*). En este paso, el atacante ha buscado un acceso remoto al sistema para crear un *shell inversa* utilizando el comando *ncat*.

iii. Descarga del *exploit*: El atacante descarga el exploit mediante *wget*.

iv. Explotar el *kernel* de *Linux 2.6.37* para ser *root*: Este exploit tiene asignada tres vulnerabilidades: *CVE 2010-4258, CVE 2010-3849, CVE 2010-3850*. Al ejecutar el *exploit*, el atacante consigue ser *root*.

v. Instalar un acceso permanente: El atacante quiere mantener el acceso como *root* permanentemente y borrar sus huellas:

A. Crea un usuario (*UID 0, sudo* y un *Gateway root* con el comando *root-sh*)

B. Ejecuta un demonio como una *shell root* (aprovechando una puerta trasera).

C. Aplica *rootkit* a nivel de *kernel* para que el atacante tenga un acceso invisible a *shell*. Finalmente, el atacante crea un nuevo usuario sobre la máquina objetivo.

4. El segundo escenario de ejemplo utilizado en [YF07] se corresponde con un ataque de *Remote to Local* (R2L) y *Local to Root* (L2R). La red LAN tiene dos *hosts*. El primero de ellos es un servidor FTP, donde el servicio *ssh* está ejecutándose y el segundo es un servidor de base de datos, el cual tiene ejecutando el servicio FTP. El objetivo del atacante es comprometer la información almacenada en la base de datos.

   (a) La primera acción es un ataque *Remote-To-Root* llamado *Sshd Buffer Overflow (SBO)*. Este ataque da a un usuario remoto una *shell root* sobre la máquina vulnerable. Esta acción se realiza sobre el primer servidor, donde el atacante debe de tener una cuenta de usuario y el *host* tiene una versión vulnerable de sshd.

   (b) La segunda acción llamada ataque ftp-rhost (ftpR), explota una vulnerabilidad de ftp que permite al intruso crear un fichero *.rhosts* en el directorio *home* de FTP el cual establece una relación de confianza entre el inicio de sesión remoto entre el host origen y destino.

   (c) La tercera acción es un ataque de inicio de sesión remoto (*rlogin*). En este ataque el intruso se registra en el host destino mediante una relación de inicio de sesión remoto de confianza.

   (d) El último paso es un ataque de *Buffer Overflow Local (LBO)*. El intruso ya tiene privilegio de usuario en el *host* objetivo y gana privilegios de *root* explotando la vulnerabilidad de *buffer overflow* de *setuid*.

5. Denegación de Servicio Distribuida (DDoS). En DARPA 2000 [DAR00] han propuesto distintas fases de ataque de DDoS basados en ataques reales en la red de las *Fuerzas Armadas del Aire*. Ambos ataques se corresponden con una Denegación de Servicio Distribuida realizada por un atacante relativamente novel. Usando un ataque multi-paso es capaz de acceder a varios host de Internet e instalarle los componentes necesarios para llevar a cabo una DDoS a un sitio gubernamental de los Estados Unidos. Como parte del ataque se usa un *exploit* para el demonio *sadmind* de las máquinas *Solaris*, un ataque bien conocido de *Remote-To-Root* para ganar acceso de *root* sobre

tres máquinas *Solaris* de una base de las *Fuerzas Aéreas*. El atacante realiza la Denegación de Servicio Distribuida utilizando la herramienta *Mstream DDoS*. Esta herramienta no es muy sofisticada debido a que no hace uso de cifrado y no ofrece un gran rango de acciones de ataque. La parte que actúa como servidor de *Mstream* es instalado en tres máquinas *Solaris* vulnerables para generar y enviar los paquetes de ataque para la denegación de servicio. La parte master de *Mstream* que es el software que proporciona la interfaz de usuario y el control de los servidores es instalado en una de las máquinas víctimas.

(a) LLDOS1.0: Incluye un ataque de Denegación de Servicio Distribuida realizada por un atacante principiante. Durante el ataque se prueba la red, se accede a máquinas con la vulnerabilidad *sadmind* de *Solaris*, se instala el troyano *Mstream DDoS* y finalmente se realiza la denegación de servicio utilizando las máquinas comprometidas sobre un servidor de la red interna.

   i. *IPsweep* de la red desde un lugar remoto para determinar los host que están en la red.

   ii. Escaneo de IPs para buscar el demonio *sadmind* ejecutado en máquinas *Solaris*.

   iii. Explotar la vulnerabilidad del demonio *sadmind* mediante un ataque de *Remote to Root*, en concreto un *Remote Buffer Overflow*, realizando intentos en todas las máquinas se tenga o no éxito.

   iv. Instalación del troyano, *Mstream DDoS software*, sobre las tres máquinas vulnerables de la red atacada. Para ello copia el software mediante *RCP*. Además crea el fichero *.rhosts* para poder hacer *rsh* para ejecutar el master.

   v. Ejecución de la Denegación de Servicio Distribuida desde todas las máquinas vulneradas hacia el servidor de la red interna objetivo. En concreto se pretende indundar a la víctima con *ACKs* de *TCP* a muchos puertos distintos.

(b) LLDOS 2.0.2: Incluye un ataque de Denegación de Servicio Distribuida de un atacante más cauteloso que el del escenario anterior, aunque todavía es un princi-

piante ya que es un ataque que cualquier persona es capaz de ejecutar y descargar. Durante las fases de ataque, se realiza un escaneo de la red, se accede a los *hosts* explotando la vulnerabilidad *sadmind*, se instala el troyano *Mstream DDoS* y se ejecuta la denegación de servicio sobre un servidor interno de la red utilizando las máquinas comprometidas.

i. Se realiza una consulta al DNS público (*HINFO*) para saber el sistema operativo de cada host de la red . Además de esta manera es capaz de saber el puntero de la pila necesario para realizar el *Buffer Overflow* de las máquinas con *sadmind*.

ii. El atacante accede al servidor de DNS usando el *exploit* de *sadmind*, pero sin realizar tantos intentos como en el caso de *LLDOS1.0* ya que se sabe el puntero de la pila.

iii. Accede a un *host Solaris* mediante FTP para incluir el software *Mstream DDoS*, incluido el master.

iv. Mediante *Telnet* inicia el master y lanza un script para buscar otros *hosts Solaris* con *sadmind* para acceder mediante *Buffer Overflow*.

v. Se ejecuta la Denegación de Servicio Distribuida vía *telnet login* sobre el master con el objetivo de inundar a la víctima mediante *TCP*.

6. En [Col13] proponen un escenario de ataque multi-paso basado en la red representada en la Figura 2.5.3 La secuencia de pasos ejecutados por el atacante es el siguiente:

(a) El atacante explota una vulnerabilidad sobre el servidor web y gana privilegios de ejecución sobre el servidor web.

(b) Los controles de acceso en el *firewall* que separan la *DMZ* con la *LAN* interna permiten el acceso al servidor de archivos desde el servidor web. El atacante utiliza el privilegio de ejecución obtenido en el servidor web para vulnerar el servidor de archivos y obtener acceso root a dicho servidor.

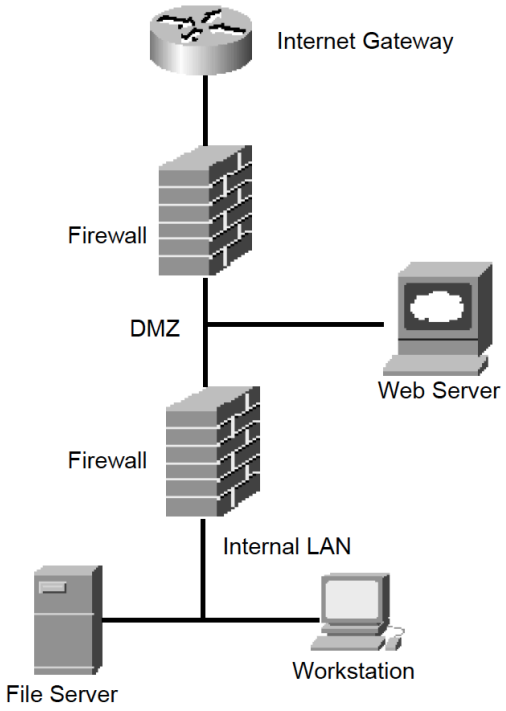


Figura 2.4.1: Red de ejemplo [Col13]

(c) El atacante usa el servidor de ficheros para modificar ficheros ejecutables los cuales más tarde serán ejecutados sobre el *host* de trabajo.

Paso

7. En [LCV03] distribuye el ataque final en sub-ataques los cuales siguen una serie de pasos que se consiguen a través de una serie de fases. Este ataque se basa en el escenario de la Figura 2.4.2 .

   (a) Ataque 1: A través del servidor Web

      i. Paso 1: Tiene como objetivo obtener acceso remoto administrativo al servidor Web y obtener todos los nombres de usuario y contraseñas disponibles.

         A. El atacante prueba el sitio Web para la vulnerabilidad *IIS Unicode* para intentar explotar un *bug* y ganar un listado de directorio. Existe una carpeta con permisos asignados a usuarios anónimos (esto existe por defecto).

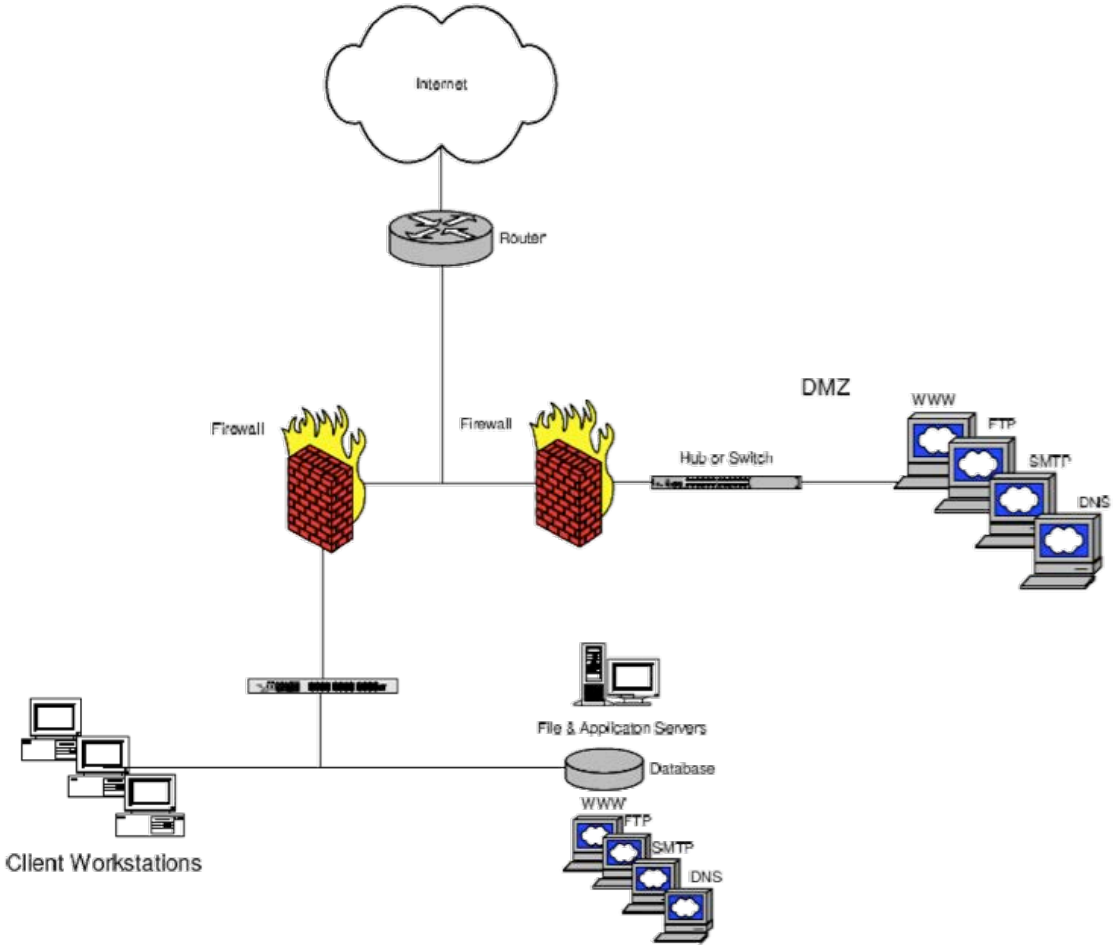


Figura 2.4.2: Red de ejemplo [LCV03]

Los usuarios anónimos tiene permisos de ejecución para cmd.exe y otros comandos del sistema operativo. Por ejemplo, *Windows 2000* con *IIS5* y *service pack 1* es vulnerable. El log de IIS detectará peticiones *http* mal formadas. (http://address.of.iis5.system/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir)

B. Una vez que el atacante sabe que el servidor Web es vulnerable, debe redirigir la salida estándar usando el caracter >, pero primero el atacante debe copiar *cmd.exe* a otro fichero. El *log* de *IIS* detectará peticiones *http* mal formadas. (http://address.of.iis5.system/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+copy+c:/winnt/system32/cmd.exe+

cmd1.exe)

C. El atacante carga herramientas mediante la creación de un archivo *ASP* personalizado con comandos *echo*. El log de *IIS* detectará peticiones http mal formadas y se verá el fichero *ASP* que se ha generado. Por ejemplo, (http://address.of.iis5.system/scripts/cmd1.exe?/c+echo+This+is+text+to+include+in+a+file+>>+file.asp)

D. El atacante se desplaza al fichero creado y realiza la carga. Los nuevos ficheros cargados aparecen en la máquina y el registro de *log* de *IIS* mostrará una petición *POST http*.

E. El atacante ejecuta *PipeUpAdmin.exe* para explotar la vulnerabilidad, consiguiendo ser usuario administrador. El *log* de *IIS* mostrará peticiones http mal formadas, se arrancará un servicio y la cuenta de usuario estará en el grupo de administradores.

F. El atacante tiene como objetivo obtener una interfaz donde poder ejecutar comandos, para ello hay varias opciones:

- *GUI* completa conseguida utilizando el cliente *VNC*. Si un usuario inicia sesión en la consola, el atacante obtiene dicha pantalla. Se podrá ver en el servidor que se ha creado un servicio asociado a *VNC* y ejecutables, por otro lado, también se detectarán peticiones http mal formadas.

- *shell* interactiva: *Netcat* podría ser usado junto con cmd.exe para conseguir una interfaz de línea de comandos. Se detectará una petición http mal formada y se visualizará el proceso de *netcat*. Por ejemplo, http://address.of.iis5.system/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+nc.exe+-L+-e+cmd.exe+-p+1234
- *shell* no interactiva: Usando el exploit emphUnicode se puede conseguir una *shell* no interactiva, la salida de los comandos ejecutados serán redireccionados a un fichero que puede ser accedido mediante peticio-

nes nomales de *GET http*. Se observarán peticiones http mal formadas.

G. *Lsadump2.exe* se ejecuta para intentar extraer servicios y contraseñas asociadas desde el *LSA*. Se podrán ver las consultas a *LSA*.

H. Las claves de registro específicas se consultan para determinar los nombres de usuario asociados a los servicios del punto anterior mediante regdmp.exe. Las claves específicas para la lectura del registro pueden ser observadas. Un ejemplo de comando,regdmp HKEY_LOCAL_MACHINE\system\currentcontrolset\services\clipsrv

I. *Pwdump2.exe* se ejecuta para extraer nombres de usuario y contraseñas *hash* desde el *SAM*. Este paso no será necesario si el acceso administrador de dominio es obtenido en el paso anterior. La salida de *pwdump2.exe* se utilizará como entrada de *Lophtcrack* para obtener las contraseñas en texto claro. Las consultas podrán ser observadas.

ii. Paso 2: Tiene como objetivo modificar el sitio Web, para ello debe ganar acceso a escritura de los ficheros de contenido de dicha Web.

A. Conocer los usuarios y grupos en el servidor de archivos y controlador de dominio. Para ello se utiliza *winfo.exe,* el cual haría que los nombres de usuarios se vieran en texto claro por la red desde el servidor web.

B. Existe un grupo que se llama *webmaster* que asumimos que tienen permisos de lectura y escritura. El atacante encuentra algunas contraseñas de cuentas del grupo *webmaster* con la ejecución de *pwdump2* y *lsadump2*. De esta manera el atacante comparte de manera remota el contenido de la web. (net use x: \\fileserver\webshare)

C. El contenido Web es modificado por el atacante. Para detectar la modificación de la página web existen herramientas específicas de monitorización de contenido Web.

iii. Paso 3: El objetivo es conseguir información de tarjetas de crédito almacenadas en la base de datos mediante consulta a la base de datos con una cuenta

con privilegios.

A. Se realiza la conexión a la base de datos *SQL* mediante *sql.exe* desde el servidor Web.

B. El atacante busca en la base de datos donde está la información de tarjetas de crédito. Las consultas *SQL* realizadas se podrán ver en la red.

(b) Ataque 2: Ataque a información privilegiada (*Insider Attack*). En este caso el atacante tiene una cuenta de usuario pero sin privilegios para acceder a la información que desea obtener. Por otro lado tampoco tiene permiso de administrador en su propia máquina.

i. Paso 1: Conseguir privilegios de administrador mirando usuarios y contraseñas de cuentas de distintos servicios ejecutados sobre el host local.

A. El atacante consigue privilegios de administrador para su máquina local mediante el *exploit PipeUpAdmin*.

B. El atacante ejecuta *lsadump2* y encuentra contraseñas asociadas a las cuentas de los servicios que están ejecutándose en la máquina local.

C. Se consulta en las Claves de Registro Específico (*Specific Registry Keys*) los nombres de usuario asociados con los servicios de los que tenemos la contraseña. Esto se realiza mediante *regdmp.exe*. Tras ello conseguimos privilegios de administrador de dominio. Un ejemplo de comando:regdmp HKEY_LOCAL_MACHINE\system\currentcontrolset\services\clipsrv.

ii. Paso 2: Conseguir acceso de escritura al fichero de contenido Web para modificar el sitio Web.

A. Aprovechando los privilegios de administrador de dominio, el atacante monta el drive por defecto en el servidor para conseguir acceso completo a toda la partición del disco. (net use x: \\fileserver\c$)

B. El atacante analiza los ficheros del contenido de la web que consulta la base de datos.

C. El atacante puede modificar el contenido de la web. Para detectar estas modificaciones habría que tener una herramienta que monitorizara el contenido de la Web.

iii. Paso 3: Conseguir información de la tarjeta de crédito de los consumidores mediante cuentas con acceso a los datos y *crackear* los *hashes* de las contraseñas para las cuentas encontradas.

A. El atacante mirará usuarios y grupos sobre la base de datos para identificar cuentas con acceso a los datos que se quieren visualizar. Para ello se utiliza *winfo.exe*.

B. El atacante localizará un controlador de dominio (*PDC*), identificando la máquina por la que él fue autenticado. Logra esto mirando las variables de entorno que contiene el controlador de dominio con el comando *set*.

C. El atacante extrae nombres de usuario y hashes de contraseñas desde el *PDC* usando *Lophtcrack* y *crackea* las contraseñas de las cuentas que fueron identificadas para tener acceso a la base de datos.

D. En este punto el atacante puede conectarse a la base de datos usando *sql.exe*. El atacante puede realizar múltiples intentos de conexión con distintas cuentas para conseguir acceso a los datos deseados, en este caso la información de tarjetas de crédito. Se observarán los intentos de acceso a la base de datos con las distintas cuentas.

E. El atacante realiza las consultas a la base de datos para conseguir la información de las tarjetas de crédito.

(c) Ataque 3: Troyano vía e-mail

i. Paso 1: Acceso desde fuera mediante el uso de una *shell* remota. La puerta trasera es proporcionada por un Troyano enviado a un correo electrónico.

A. El atacante hace un Troyano usando una aplicación llamada *elitewrap* que envuelve el juego con otro programa llamado *Ackcmd* en un paquete binario para que se ejecuten juntos. Cuando se está jugando el *Ackcmd* se

ejecuta sin que el usuario tenga conocimiento y permite la ejecución de programas remotos usando un *TCP ACK-tunnel* para evitar los *firewalls*. *Ackcmd* continúa ejecutándose aunque se borre la aplicación. Este Troyano se envía por email a un empleado. En este caso se podría observar tráfico *SMTP* llevando un ejecutable en los datos adjuntos. También podría verse el ejecutable en el servidor de correo. Cuando se ejecuta el Troyano el *Ackcmd* podría verse en la lista de procesos de la máquina, aunque puede ser renombrado para parecer un proceso del sistema.

B. El atacante puede conectarse a la máquina a través del *firewall* usando el cliente de *Ackcmd* con los permisos de la persona que está utilizando el juego. La conexión simula ser una sesión normal de *http*.

C. El atacante puede descargar otras herramientas mediante FTP a la víctima.

(d) Ataque 4: Ataque a través de un usuario que está en un ordenador de su casa a través de una conexión VPN con permisos de administrador de dominio.

i. Paso 1: Atacar a la cuenta de administrador del ordenador de la casa para obtener información de la configuración y autenticación de la VPN.

A. El atacante identifica al administrador del ordenador de la casa que está conectado a Internet a través de un cable al modem. Esto se puede hacer realizando un sniffing de paquetes sobre el modem usando el navegador *Netbios*. Además el atacante ahora tiene la dirección *IP* de la máquina objetivo. Al ser una actividad pasiva es muy difícil de detectar, sin embargo se puede utilizar un sensor para detectar una interfaz de red en modo promiscuo.

B. El atacante enumera las acciones en el pc utilizando el comando *net* con el argumento *view*. Se podrían observar conexiones *SMB* en la máquina víctima.

C. El atacante consigue acceso de lectura y escritura al *drive*. Se podría ver en

la red una conexión *SMB* remota.

D. Después de montar el *drive,* el atacante determina el software VPN y la configuración que se está usando. Se podrá ver actividad de disco y tráfico *SMB* en la máquina víctima.

E. Encontrar el usuario y contraseña para conectarse a los recursos corporativos. La primera opción es buscar un fichero de texto con los datos de atenticación y si no estuviera utilizar un *keyboard logger* para obtener dicha información. En este último caso se vería un proceso en la lista de tareas del ordenador. Además el fichero keystores que genera el programa podría ser visible. Este fichero se transmite vía *SMB* y puede verse en la red.

ii. Paso 2: Configurar la máquina del atacante para imitar el comportamiento del ordenador de casa del administrador y establecer un túnel VPN con la organización.

A. El atacante instala el *software VPN* correspondiente en su máquina y se conecta utilizando el usuario y contraseña encontrado en el paso anterior. De esta manera accede a la red corporativa atravesando el *firewall* con privilegios de administrador de dominio. Además este acceso no será detectado como anómalo.

iii. Paso 3: Determinar la localización del contenido Web y modificarlo.

A. El atacante usa la herramienta de administración *ID* para examinar la configuración del servidor *Web IIS* y determinar dónde está almacenado el contenido de la web. La dirección *IP* de la máquina puede ser recuperada a través de peticiones estándar de DNS. Desde el servidor web se vería una conexión *SMB*.

B. El atacante monta el sistema de ficheros con el contenido de la web y determina el nombre de la base de datos y tablas usadas para la aplicación web. El atacante puede localizar fácilmente esta información mediante

una búsqueda de texto para una palabra clave específica en la base de datos como por ejemplo DNS. Se detectaría tráfico SMB y actividad de disco en el servidor Web.

C. En el caso de que el atacante tenga permisos de lectura y escritura sobre el contenido Web, el contenido podría ser modificado. En el caso contrario, debe usar su privilegio de administrador de dominio para montar la unidad para conseguir el acceso necesario y así poder modificar la Web.

iv. Paso 4: Conseguir la información de las tarjetas de crédito de la base de datos.

A. El atacante se conecta a la base de datos utilizando la herramienta de administración *MSSQL* con el usuario y contraseña del administrador de dominio. Visualizándose el tráfico *TCP* correspondiente.

B. El atacante mira los permisos para acceder a la información de tarjetas de crédito usando la herramienta de administración. En el caso de que no tenga los privilegios adecuados, hace que su cuenta forme parte del grupo que tenga acceso a los datos.

(e) Ataque 5a: Es como el *Ataque 4* pero asociado a plataforma *Unix*.

i. Paso 1: Obtener una *shell root* sobre el servidor web.

A. El atacante consigue una *shell* root con el *exploit Bind* sobre el servidor de DNS en la *DMZ*. Esta invocación podría ser detectada por un *IDS*.

B. El atacante obtiene la contraseña de *root* a través del fichero *shadow FTP* de manera *offline* utilizando una herramienta de fuerza bruta como *crack* o *John*. Se podría ver como el fichero es transferido desde el servidor de DNS a través de Internet, o monitorizando, se puede detectar el acceso al fichero *shadow*.

C. El servidor web es configurado para usar la misma contraseña de *root* que el servidor de DNS. Así que el atacante puede realizar un *ssh* al servidor web y examinar los ficheros de configuración de *Apache* para determinar la localización del contenido web. Desde la red se puede ver la conexión

ssh desde el DNS al servidor web.

ii. Paso 2: Reemplazar el contenido web con el contenido que el atacante quiera reubicando la web.

A. El atacante analiza el contenido de la web para determinar la base de datos y el nombre de las tablas; además de usuarios y contraseñas para la base de datos. El atacante gana información habilitando el acceso remoto a la base de datos. Si se monitorizaran los patrones de acceso a los ficheros del contenido web, se podría ver alguna anomalía.

B. El atacante mediante FTP añade nuevo contenido a la web utilizando el servidor FTP. El nuevo contenido web se pone en un directorio nuevo y se restaura el demonio *Apache*; esta actividad podría ser detectada. También se detectaría la sesión FTP para obtener el nuevo contenido, la restauración del *Apache* se notificaría en el fichero de l*og* y se verían nuevos ficheros en el servidor web.

iii. Paso 3: Obtener la información de las tarjetas de crédito mediante la creación de una nueva página web que consulta la base de datos y devuelva la información de dichas tarjetas.

A. El atacante crea un fichero HTML que consulta la base de datos y retorna la información de las tarjetas de crédito. Esto se puede realizar en el mismo servidor o transferirlo con el resto del contenido web una vez creado. Esta consulta posiblemente sea inusual, por lo que podría ser detectada, además se podría ver una petición *http* correcta para una página web desconocida.

(f) Ataque 5b:

i. Paso 1: Obtener una *shell root* sobre el servidor web.

A. El atacante explota ssh sobre el servidor web para conectarse con la cuenta root. El exploit se podría ver en el tráfico de red y los ficheros de log mostrarían una operación anómala de *ssh*.

B. Paso2: En este punto se continúa desde el Paso2 del *ataque 5a*.

En las tablas Tabla 2.4.1, Tabla 2.4.2 y Tabla 2.4.3 muestran todos los escenarios de ataque propuestos en trabajos previos, excepto el escenario 7(b) ya que es igual que el ataque 7(a) pero llevado a cabo desde un *insider*, el 7(e) y 7(f) ya que es el mismo ataque que el mostrado en 7(d) pero para Unix y los escenarios 5(a) y 5(b) correspondientes a los DDoS de DARPA 2000 debido a que se utilizan para la validación de una de las contribuciones de la tesis. Cada una de las columnas se corresponden con fases de ataque típicas de cualquier tipo de ataque:

1. Escaneo de la red de la organización para averiguar las distintas IPs accesibles, los puertos que tienen abiertos y los servicios disponibles en cada una de las máquinas para el análisis posterior de vulnerabilidades.

2. La mayoría de los ataques requieren el acceso remoto a una máquina con un usuario válido a no ser que sea realizado por un empleado de la propia organización (*insiders*). Este acceso se obtiene con un usuario sin privilegios de root o administrador por lo que no podría realizar la ejecución de comandos o scripts de ataque.

3. En la gran parte de los casos tener acceso como root o administrador a la máquina a atacar es fundamental para poder llevar a cabo el objetivo final del ataque, de manera que se pueda obtener información privilegiada o ejecutar scripts de ataque.

4. A partir de este momento el atacante puede llevar a cabo el ataque correspondiente.

Estas fases típicas son necesarias para cada una de las máquinas implicadas en el ataque, ya que el atacante no siempre tiene acceso directo a la máquina que quiere atacar. Dentro de cada una de estas fases se puede ver que cada ataque multi-paso tiene sus propias sub-fases específicas de dicho ataque.

En general, un escenario de ataque está compuesto de distintos pasos de ataque relacionados que pretenden conseguir el mismo objetivo. La diferenciación de estos pasos, debe

Escenario	Objetivo	Esceneo	Acceso a la máquina	Acceso root a la máquina	Ejecución del ataque
1	*WannaCry*	Buscar máquinas vulnerables a SMB EternalBlue	Explotar la vulnerabilidad SMB EternalBlue (*CVE 2017-0145*)		Instalar el *ransomware* WannaCry
2	*Local to Root*(L2R)	No es necesario ya que es usuario local de la máquina a comprometer		Crea una copia de *csh* en el archivo de mail de root Falsifica el bit *setuid* del archivo de correo Crea y envía un mensaje vacío a root vía la herramienta mail Ejecuta el mail de root para ganar acceso a una *shell* con privilegios de root	Ya tiene acceso al sistema por lo que puede realizar cualquier ataque
3(a)	*Remote to Local* (R2L)	Búsqueda de máquinas activas Escaneo de puertos y búsqueda de servicios disponibles	Aplica fuerza bruta para obtener usuarios y contraseñas	Consigue contraseña administrador con la herramienta SkipFish Crear puerta trasera en el servidor Apache Creación de una *shell* inversa	Ya tiene acceso al sistema por lo que puede realizar cualquier ataque
3(b)	*Remote to Local* (R2L) *User to Root (U2R)*	Búsqueda de máquinas activas con nmap Uso de la herramienta SkipFish para búsqueda de vulnerabilidades	Explota la vulnerabilidad del foro phpBB22 2.0.10 CVE 2005-2086 Creación de la *shell* inversa Descargar exploit con wget	Explotar el Kernel de Linux para ser root Instalar acceso permanente a root usando un rootkit	Ya tiene acceso al sistema por lo que puede realizar cualquier ataque

Tabla 2.4.1: Escenarios de ataque propuestos en trabajos previos

Escenario	Objetivo	Esceneo	Acceso a la máquina	Acceso root a la máquina	Ejecución del ataque
4	*Local to Root*(L2R)	No es necesario ya que es usuario local de la máquina a comprometer		sshd Buffer Overflow	Ataque ftp-rhost (ftpR) para establecer una relación de confianza con la máquina destino del ataque
	Remote to Local (R2L)	Ya sabe que máquina quiere atacar	Realiza el inicio de sesión remoto mediante la relación de confianza realizada	Buffer Overflow Local de setuid	Ya tiene acceso al sistema por lo que puede realizar cualquier ataque
6		Busca vulnerabilidad del servidor web organización	Explota vulnerabilidad del servidor web	Gana privilegios de root	Usando los privilegios de ejecución puede vulnerar otras máquinas
		Busca acceso a una máquina de la LAN interna	Acceso al servidor de archivos	Vulnerar servidor de archivos para obtener acceso a root	Modificación de ficheros ejecutables
7(a)	*Ataque al servidor web*	Comprueba si el sitio web tiene la vulnerabilidad IIS Unicode	Redirige la salida estándar Cargar herramientas con un archivo ASP personalizado Carga de ficheros Explota la vulnerabilidad IIS Unicode ejecutando PipeUpAdmin.exe	Explota la vulnerabilidad IIS Unicode ejecutando PipeUpAdmin.exe Utiliza el cliente VNC o Necat + cmd.exe o Exploit emphUnicode	Extraer servicios y contraseñas con Lsadump2.exe Consular claves específicas de registro asociados a los servicios mediante regdmp.exe Pwdump2.exe extrae nombres de usuario y contraseñas hash desde SAM + Lophtcrack para obtener contraseñas en claro Buscar usuarios con permisos de escritura en el sitio web mediante winfo.exe + pwdump2 + lsadump2 Buscar información de tarjetas de crédito en la base de datos

Tabla 2.4.2: Escenarios de ataque propuestos en trabajos previos

Escenario	Objetivo	Esceneo	Acceso a la máquina	Acceso root a la máquina	Ejecución del ataque
7(c)	*Troyano vía email*	Ya sabe que máquina quiere atacar	Crear puerta trasera a partir de un Troyano (Ackcmd) enviado por correo electrónico	Depende del usuario atacado accederá con o sin privilegios	Puede realizar algún ataque a la máquina comprometida
7(d)	*Acceso a través de la VPN de un empleado*	Identificar IP de la máquina que realiza la conexión VPN mediante sniffing de paquetes usando Netbios Identificar el usuario administrador	Enumera las acciones del pc con el comando net y el argumento view Consigue acceso de lectura/escritura al drive		Determinar el software VPN y su configuración Encontrar usuario/contraseña VPN mediante keyboard logger
		Ya sabe que máquina quiere atacar	Configurar máquina atacante para imitar el ordenador atacado y establecer tunel VPN (tenemos ya usuario administrador)		Examina la IP de l servidor web mediante peticiones DNS Localiza el contenido de la web y de la base de datos Si no lo tiene conseguir permisos de lectura/escritura y conexión con la base de datos Buscar información de tarjetas de crédito en la base de datos

Tabla 2.4.3: Escenarios de ataque propuestos en trabajos previos

ser realizado en función de las distintas unidades que puede detectar los distintos sensores de alertas y logs. Además, a la hora de diseñar un modelo de pasos de ataque es fundamental estudiar la especificidad o generalidad que deseamos que tenga el modelo.

2.4.1 Amenaza Persistente Avanzada

Uno de los ejemplos de ataque multi-paso son los ataques denominados APT (*Advanced Persistent Threat*). La Amenaza Persistente Avanzada es una amenaza que surge como evolución de las propias organizaciones debido que cada vez más la información es alcanzable a través de la red [Col12]. El objetivo de los APT es la extracción de información confidencial y el acceso durante largos periodos de tiempo hasta comprometer la entidad. Esta característica de *persistencia* es uno de los motivos que lo hace diferente al resto ataques tradicionales y por ese motivo esta tesis doctoral lo incluye como una sección a parte del resto de ataques. Estos periodos de tiempo pueden estar en el orden de meses, donde el atacante persiste en la red para obtener información de la organización. En la gran mayoría de los casos se pretende conseguir acceso desde el exterior a través de un adjunto en un correo o que un usuario pinche en un enlace, para que una vez dentro de la red, aplicar métodos más sofisticados hasta conseguir la información sensible con un fin económico o político.

La Amenaza Persistente Avanzada está enfocada a múltiples fases, siendo cada unos de los ataques de APT únicos y diferentes por lo que hay muchas variaciones, pero todas tienen en común unas fases generales [Col12]:

- *Fase 1 - Reconocimiento:* Recoge información sobre el objetivo, buscando por áreas específicas que pueden ser enfocadas a lograr un compromiso a largo plazo con el mínimo esfuerzo. Normalmente se basa en encontrar un individuo que pueda ser usado para la fase 2.

- *Fase2 - Intrusión inicial:* Determinando y encontrando algún camino hacia la organización para establecer un punto de apoyo. Este paso normalmente no requiere la explotación de alguna vulnerabilidad y en la mayoría de los casos se consigue convenciendo a algún usuario a abrir un adjunto o a pulsar un enlace.

- *Fase3 - Establecer una puerta trasera:* Las APT quieren ser capaces de comunicarse con la red objetivo. Tras la intrusión inicial, se establece un camino remoto para que el atacante continúe dentro de la red comprometida.
- *Fase 4 - Obtención de credenciales:* El atacante quiere acceder a la red entera y mantener acceso a largo plazo para su uso actual y futuro. Esto normalmente requiere obtener credenciales privilegiadas.
- *Fase 5 - Instalar herramientas:* En este punto el atacante quiere establecer persistencia y total control sobre la red. Esto normalmente se hace mediante la instalación de herramientas personalizadas para crear una comunicación command and control (C&C) con la red comprometida.
- *Fase 6 - Exfiltración de datos:* La etapa final es tratar de extraer información crítica de la red de manera cautelosa. Para esta fase se utiliza cifrado y enmascaramiento de datos para que parezca tráfico legítimo.

Este tipo de escenario se ha utilizado para la realización de una prueba de concepto llevadas a cabo dentro de un proyecto con el Mando Conjunto de Ciberdefensa (véase Tabla 7.4.1). En este proyecto se hace uso de dos tipos de sistemas de detección de intrusiones, Suricata como IDS de red y Oseec como IDS de host. A continuación se muestran las fases llevadas a cabo:

1. Ataque al navegador usando el *framework BeEF* explotando el navegador del usuario. La alarma de Suricata se corresponde con "Possible BeEF module in use".
2. Se establece una puerta trasera a partir de un *Reverse* shell. Hay distintas alarmas que proporciona Suricata: "ET INFO JAVA - ClassID" o "ET INFO JAVA .jar request to dotted-quad domain" o "ET INFO JAVA - Java Archive Download"
3. Posteriormente se realiza un escalado de privilegios vía bypassUAC para poder obtener las passwords de los usuarios (hashdump). Ossec reporta una alarma del tipo "Successful sudo to ROOT executed".

4. La exfiltración de información reporta una alarma en Suricata, "Possible information leak".

5. Para conseguir persistencia se añade un usuario nuevo llamado "hacker1" al grupo de administradores donde se obtiene una alarma desde Ossec, "User account enabled or created"

2.5 Sistemas de detección y respuesta a intrusiones

La tesis doctoral tiene por objetivo el diseño y desarrollo de una solución centrada en la defensa perimetral de la red de una organización. A continuación se va a mencionar en mayor detalle las características de algunos sistemas de defensa perimetral enfocados en la detección y respuesta a intrusiones para analizar las ventajas e inconvenientes sobre cada uno de ellos con el objetivo de mantener a salvo los distintos activos de una organización

2.5.1 Sistemas de Detección de Intrusiones

Los Sistemas de Detección de Intrusiones se encargan de monitorizar eventos en busca de comportamientos maliciosos o inesperados. Los IDSs se pueden clasificar según una serie de características [Amo99]:

1. Según el tipo de fuentes de información que monitoricen:

   (a) HIDS (*Host-based* IDS): Recopila información interna de la máquina, como actividad del usuario, sistema operativo o ficheros del sistema, en busca de comportamiento anómalo. Monitoriza un único ordenador por lo que la carga procesada es mucho menor, pero su gestión en redes con muchos sistemas se vuelve muy compleja.

   (b) NIDS (*Network-based* IDS): Recopila información del tráfico de la red en modo promiscuo, intentando identificar patrones de tráfico malicioso. Este tipo de IDS es transparente para la red e invisible para los atacantes, pero tienen que procesar mucha información en redes con mucha carga de tráfico.

(c) NNIDS (*Network-Node* IDS): Monitoriza la información de tráfico local, dirigido o generado por un único nodo.

(d) AIDS (*Application-based* IDS): Se encarga solo de monitorizar la información de una aplicación concreta de un sistema, detectando así intrusiones específicas de la aplicación.

2. Según la frecuencia de tratamiento de eventos:

   (a) IDSs que realizan un análisis periódico del tráfico de red. (No muy usual)

   (b) IDSs que realizan una monitorización continua de eventos y simultáneamente realizan el análisis. (Uso más común)

3. Según los principios de detección

   (a) Detección de firmas: Se basa en la búsqueda de analogías entre lo que ocurre en la red o sistema en un momento concreto y los patrones predefinidos que describen un ataque conocido, lo que se conoce como su firma. No detecta nuevos ataques y se debe realizar la actualización de las firmas constantemente. Tiene pocos falsos positivos, pero muchos falsos negativos.

   (b) Detección de anomalías: Se basan en la identificación de comportamiento inusual en los sistemas y redes. Para ello es necesario que el IDS aprenda dentro de un entorno controlado, libre de intrusiones, para construir los perfiles de actividad. El problema principal es que estos sistemas tienen muchos falsos positivos.

   (c) Híbridos: Un ejemplo de este tipo de IDS se muestra en [AZC09]

4. Según su estrategia de control:

   (a) Centralizada: Cada IDS es un punto de detección de eventos que son enviados a un centro de análisis común. De esta manera, todos los IDSs monitorizan datos, pero solo uno analiza.

   (b) Distribuida: Cada IDS es independiente, es decir, se encarga de la monitorización de la red y de la obtención y análisis de la información.

Generalmente, los IDSs son dispositivos pasivos, encargados de la detección de intrusiones, aunque en algunos casos, pueden llevar a cabo acciones de respuesta pasiva como la notificación de la alarma o rara vez, crear reglas en el cortafuegos de duración limitada.

Los Sistemas de Detección de Intrusiones [A+80] han evolucionado rápidamente y a día de hoy hay gran variedad de herramientas maduras basadas en distintos paradigmas según sus principios de detección.

2.5.2 Sistemas de Prevención de Intrusiones

La principal diferencia entre un Sistema de Detección de Intrusiones y un Sistema de Prevención de Intrusiones es el tipo de respuestas que puede realizar frente a una amenaza. Además de la respuesta de bloqueo, un IPS puede responder a una amenaza ejecutando respuestas básicas: reconfigurar controles de seguridad, como un cortafuegos o un router, para bloquear un ataque, aplicar parches de seguridad o eliminar contenidos nocivos de un ataque, como por ejemplo un archivo adjunto infectado de un correo electrónico antes de que el usuario reciba el correo. Los IPSs se pueden clasificar en cuatro tipos [Sta04]:

- NIPS (*Network-based Intrusion Prevention System*): Busca tráfico sospechoso en la red de la organización.
- WIPS (*Wireless Intrusion Prevention System*): Igual que un NIPS, pero busca tráfico sospechoso mediante análisis de protocolos de las redes inalámbricas.
- NBA (*Network Behavior Analysis*): Analizan el tráfico de red para identificar amenazas que generan flujos de tráfico inusuales.
- HIPS (*Host-based Intrusion Prevention System*): Este tipo de IPS es una aplicación software instalada en una máquina, para detectar actividades sospechosas mediante el análisis de acontecimientos que ocurren dentro de ese host.

El IPS debe estar ubicado en línea con el flujo de transacciones que incluyen la intrusión. Por ejemplo, un IPS que responde a intrusiones generadas por un NIDS debe estar ubicado sobre un cortafuegos [Des03]; por el contrario, un IPS que responde intrusiones generadas

por un HIDS que monitoriza llamadas al sistema, debe estar ubicado en el kernel del sistema operativo [ROC05].

2.5.3 Sistemas de Respuestas a Intrusiones

Los Sistemas de Respuesta a Intrusiones [SBW07] son los responsables de inferir y ejecutar una acción de respuesta frente a las intrusiones detectadas por otros dispositivos, como puede ser un IDS. La diferencia con los IPS e IDS es que no solo se limita a acciones de bloqueo, sino que se dispone de un amplio catálogo de acciones de respuesta que se pueden ejecutar sobre otros componentes de seguridad. Los IRS se pueden clasificar en función del tipo de respuesta proporcionado:

- Sistemas de notificación: La respuesta consiste exclusivamente en informes y alarmas para que el administrador del sistema pueda saber que ha ocurrido.
- Sistemas de respuesta manual: Además de informar al administrador del sistema, proporciona un conjunto de respuestas que pueden ser útiles para contrarrestar el ataque. La elección de una de las respuestas es tarea del administrador.
- Sistemas de respuesta automática: Sistemas que responden inmediatamente a la intrusión sin necesidad de intervención del administrador del sistema, reduciendo en gran medida el tiempo de respuesta. Estos sistemas se denominan Sistemas de Respuestas a Intrusiones Automáticos (AIRS) y se basan en una serie de métricas para la selección de la respuesta como se muestra en la Figura 2.5.1 .

Además, otro factor a tener en cuenta son los parámetros utilizados para determinar las métricas de selección de las respuestas en los sistemas autónomos. Estos datos pueden ser obtenidos de distintas fuentes:

- Valores obtenidos a partir de los IDSs: Como puede ser la confianza en el IDS y los reportes de la intrusión.
- Corporativos: Por ejemplo la importancia de los activos de la red corporativa.

Figura 2.5.1: Esquema de respuesta de un AIRS

- Contexto de la red o el sistema: Uno de ellos puede ser el nivel de actividad de la red.
- Respuesta a la intrusión: Se debe analizar el coste, la severidad, la eficacia y la eficiencia de la respuesta a ejecutar.

Además, los IRSs pueden basarse en dos tipos de políticas de respuestas:

- Respuestas reactivas: La acción de respuesta se ejecuta una vez que se ha detectado una intrusión. La confirmación de la intrusión puede basarse en métricas de confianza del IDS o en una coincidencia del vector de ataque con la firma definida en una base de datos. Por lo general, un IRS reactivo intenta minimizar el daño causado por una intrusión o restaurar el estado del sistema.
- Respuestas proactivas: Son respuestas que se lanzan antes de que una intrusión comprometa el activo objetivo del atacante. Estas respuestas están estrechamente ligadas con la capacidad de predicción, de tal manera que sea capaz de responder antes de que el atacante consiga su objetivo final.

La principal diferencia entre estos tipos de respuestas es el tiempo en el que se ejecuta la

acción de respuesta frente al marco temporal de una intrusión. En función de estos tiempos se tienen tres intervalos [APFC10]:

- Tiempo antes de la intrusión: En este intervalo de tiempo la ejecución de una respuesta proactiva previene el objetivo final del atacante.
- Tiempo después de la intrusión: En este momento se ejecuta la respuesta reactiva con el objetivo de minimizar el impacto causado por la intrusión. Este intervalo será mayor o menor en función de si el sistema de respuesta es manual o automático.
- Tiempo después de la respuesta reactiva: Este tiempo comienza justo cuando se ha ejecutado la repuesta reactiva y es el momento donde el sistema se encarga de evaluar la efectividad de las respuestas ejecutadas durante el ataque. Aquí entra en juego la capacidad de auto-aprendizaje del IRS para mejorar las futuras respuestas ante intrusiones.

Esta diferencia repercute en la ventana temporal que tiene el atacante para conseguir el objetivo final. La Figura 2.5.2 representa el marco temporal del ataque y se puede ver las distintas fases que el atacante lleva a cabo hasta conseguir la intrusión final. La primera de las respuestas que el IRS puede ejecutar se basa en la predicción del objetivo final en función de las fases del ataque, y se corresponde con la respuesta proactiva, en este caso la respuesta no es efectiva y el sistema no intenta ninguna otra respuesta de este tipo. De este modo, el atacante continuaría sus fases hasta conseguir el objetivo final de la intrusión el cuál es detectado en la segunda línea vertical del esquema. Una vez detectada la intrusión el IRS ejecutaría una respuesta reactiva para intentar restablecer el estado normal del sistema.

Por otro lado, los IRSs se pueden clasificar en función de su capacidad de adaptación en:

- Estáticos: El mecanismo de selección de la respuesta se mantiene invariante a lo largo del tiempo.
- Adaptativos: Son los sistemas que son capaces de ajustar la respuesta de manera dinámica. En el caso de un AIRS adaptativo, la respuesta se adaptaría automáticamente en función de determinados factores, como por ejemplo, en la propuesta de tesis, con la

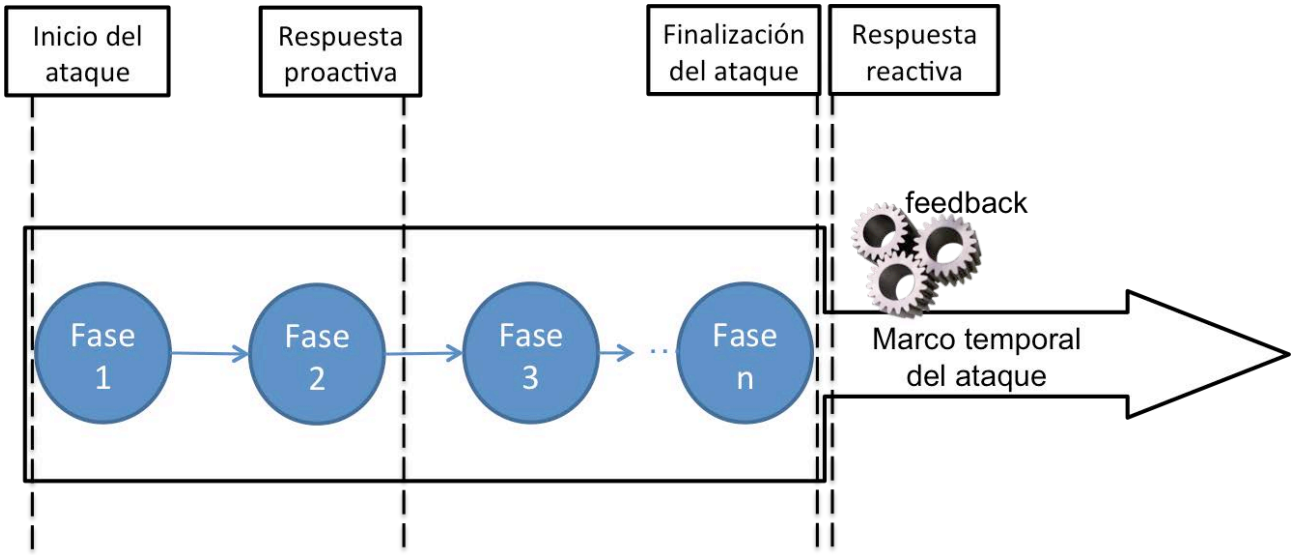


Figura 2.5.2: Relación entre tipos de respuestas y el marco temporal de un ataque

efectividad de la respuesta. En este sentido en la Figura 2.5.2 se puede ver que el IRS evalua las respuestas lanzadas durante el marco temporal del ataque, para conseguir feedback y mejorar en ataques sucesivos.

En este apartado se ha detallado la clasificación de los IRSs según las características relevantes para esta tesis, aunque la categorización de los IRSs se puede realizar teniendo en cuenta otras propiedades que se pueden ver en [ML13]. Los IRSs son una de las tecnologías menos maduras dentro de los sistemas de defensa perimetral. Debido al aumento de ataques a los sistemas informáticos, es necesario reducir la ventana de oportunidad del atacante. Por ello es necesario mejorar la velocidad de reacción y la eficacia de las respuestas mediante la automatización, proactividad y adaptabilidad de los sistemas de respuestas.

2.5.4 Sistemas de Gestión de Eventos

Los SIEM (*Security Information and Event Management*) se basan en recogen la información desde los distintos sistemas de defensa perimetral [Dub02]. Las herramientas de sistemas de gestión de información de seguridad agregan datos y eventos de manera similar a los sistemas de gestión de red, pero aplicado a los registros de eventos generados desde distintos dispositivos de seguridad como *firewalls*, servidores *proxy*, sistemas de detección de intru-

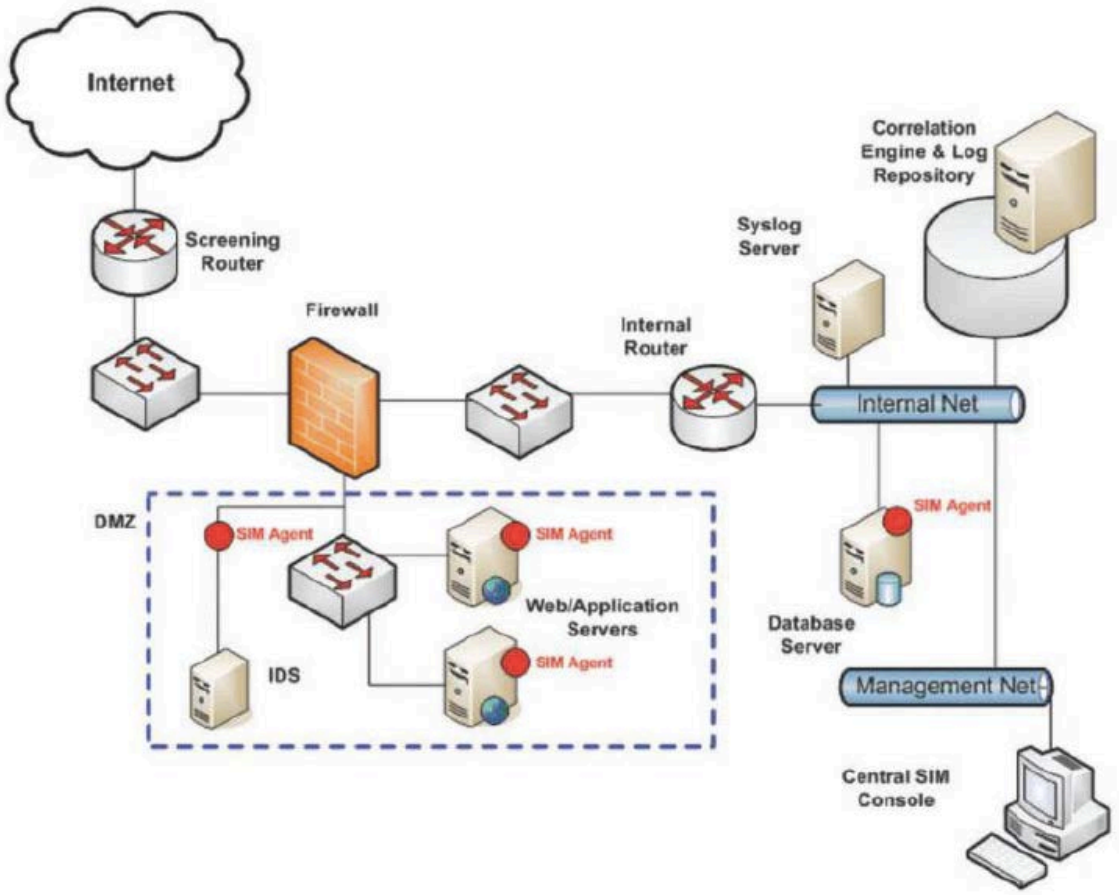


Figura 2.5.3: Ejemplo de arquitectura SIEM [Dub02]

siones y antivirus. La función más interesante de esta tecnología es la capacidad de normalizar estos registros de eventos. Por ejemplo, un SIEM puede tener registros de servidores de seguridad de *firewalls*, *routers* y *switches*, sistemas de detección de intrusiones Snort y servidores Windows; creando un formato de salida consistente. La vista de estos elementos normalizados crea una visión más completa del entorno. Algunas arquitecturas normalizan los eventos antes de ser almacenados en la base de datos, otras soluciones, lo formalizan para la presentación en la consola. Usualmente el formato de normalización de los eventos es XML.

Los agentes SIM normalmente son ejecutados en los ordenadores que se desean monitorizar. Estos envían la información a un servidor centralizado que actúa como consola de

seguridad como se puede ver en la . El envío de eventos de seguridad, que se mostraran en informes, tablas y gráficos, son a menudo en tiempo real. Los agentes pueden incorporar filtros locales para reducir la información que comunican al servidor. La consola de seguridad es supervisada por un operador de la empresa que revisa toda la información recogida y toma medidas en respuesta a las alertas emitidas. La mayoría de los productos SIEM tienen unos componentes comunes.

- Recolector de los eventos de los registros. Esto puede tomar la forma de agente dentro de un sistema como un servidor UNIX o Windows; o una aplicación basada en Web que se registra en el sistema y recoge los datos del evento. Este último es el que está ganando una mayor popularidad debido a que muchas organizaciones prefieren no hacer uso de numerosos agentes en su entorno.
- Motor de logging y correlación, el cual viene a menudo junto a un repositorio de almacenamiento grande para almacenar los datos de los eventos normalizados que el sistema SIEM recoge. Dependiendo de la arquitectura en particular, la infraestructura de almacenamiento de gran capacidad más habitual, como *Network Attached Storage* (NAS) o *Storage Area Networks* (SANs), pueden ser usados para el almacenamiento de eventos a largo plazo.
- Consola de gestión central. Tiene como características habituales la monitorización de alertas, investigación de eventos y análisis forense, elaboración de informes y la administración de los otros componentes de la arquitectura SIEM.

2.6 Análisis de investigaciones previas

En este apartado se analiza el estado del arte en sistemas de detección y respuesta, predicción de intrusiones y técnicas de correlación de eventos dentro del ámbito de investigación de la ciberseguridad. Estos estudios se basan en distintos modelos de aprendizaje para conseguir su objetivo. A continuación se detallan algunas de las investigaciones.

2.6.1 Sistemas de detección de intrusiones

Hay investigaciones sobre IDSs basados en modelos mátemáticos como por ejemplo [KZD+15], que propone un IDS basado en firmas para la detección de Denegación de Servicio Distribuida utilizando Naïve Bayesian Classifier. Por el contrario, hay casos en los que se propone el desarrollo de IDSs basado en Redes Neuronales, como es el caso de [MAHZ12] y [NAAS12] que se basan en el algoritmo de *Backpropagation* para la detección de intrusiones.

Hay varios trabajos que implementan funciones para clusterizar y correlar alertas como un modulo cooperativo del IDS y así reducir el número de alertas transmitidas. El componente de agregación y correlación (ACC, Aggregation and Correlation Component) propuesto en [DW01] crea relaciones entre eventos relacionados de acuerdo a unas reglas explícitas de *Prolog*. Cuando ocurre un evento, ACC verifica si la alerta actual es una consecuencia de alguna recibida previamente. Una propuesta similar [NCR02] revela estrategias de ataque basado en grafos de correlación. Este método identifica los prerrequisitos como la existencia de un servicio vulnerable y las consecuencias como el descubrimiento de servicios vulnerables. El problema principal de estos trabajos es la generación manual de las reglas de correlación. Para evitar esto, F. Cuppens et. al. [CM02] generan automáticamente reglas de correlación tras un proceso *off-line*. Como resultado de la correlación se obtiene un conjunto de planes candidatos que corresponden a la intrusión bajo ejecución.

2.6.2 Sistemas de respuestas a intrusiones Automático

- Dentro del ámbito del proyecto RECLAMO, se propone una arquitectura de seguridad capaz de seleccionar de manera dinámica la respuesta más apropiada [ML13]. En concreto se tienen en cuenta factores como el contexto del sistema, el coste de la respuesta, la importancia de los recursos y la eficiencia de las respuestas para lograr un sistema autónomo y adaptativo basado en la evaluación del daño causado por la intrusión y el coste de las respuestas. Este sistema se desarrolla basado en modelos formales de información definidos por ontologías [dVVM+09]. Esta ontología representa la in-

formación sobre las intrusiones, los parámetros de autoevaluación, la confianza y reputación en los IDSs, etc. Las métricas de seguridad, representadas mediante el lenguaje formal SWRL (*Semantic Web Rule Language*) [HPSB+04], utilizan esta información para inferir la respuesta más apropiada. Además, incorpora un método de evaluación de la efectividad de la respuesta basada en la Entropía de Renyi del contexto, pero tiene como inconveniente la cantidad de parámetros y umbrales que hay que determinar previamente.

- MAIM [PXG+12] es un sistema de respuesta a intrusiones adaptativo basado en la inmunidad artificial, es decir, utilizan un algoritmo bio-inspirado para determinar la efectividad del sistema. Este enfoque implementa una política de acuerdo al riesgo global del sistema, aprendiendo sobre los estados peligrosos de la red para detectar falsos positivos. MAIM lanza respuestas más o menos estrictas dependiendo de dicha política predefinida. Esto es contrario a la propuesta de esta tesis doctoral en la que se utiliza el aprendizaje bio-inspirado con el objetivo de evaluar dinámicamente la respuesta frente a un ataque concreto y el AIRS se encarga de detectar falsos positivos analizando el contexto de la red y del sistema, como se detallará en el capítulo 5.

- COSIRS [IS12] es un AIRS que basa la selección de las respuestas en tres factores: el coste del daño de la intrusión, el coste de la respuesta automática y el coste operacional. En cambio, no tiene en cuenta la efectividad que ha tenido la respuesta seleccionada contra el ataque en curso.

- *Adaptative Intrusion Tolerant Systems* (ADEPTS) [FWM+05][WFM+07], es un AIRS basado en grafos cuyo objetivo es monitorizar y rastrear intrusiones detectadas en un sistema distribuido de servicios relacionados, en especial sistemas de comercio electrónico. En concreto cosiste en una estructura en forma de grafo dirigido donde cada nodo representa un ataque y a partir de los incidentes reportados por los IDSs estima la trayectoria más probable de propagación del ataque dentro del sistema. Además tiene un mecanismo de realimentación que evalúa el éxito o fracaso de una respuesta

desplegada previamente basado en que no se detecten ataques del mismo tipo dentro de un periodo de tiempo establecido. Si no se detectan ataques durante ese tiempo se incrementa el valor correspondiente a la respuesta evaluada en una cantidad fija configurable. En caso contrario se disminuye su valor dependiendo del nodo. El problema de esta propuesta es la falta de escalabilidad del modelo, ya que el grafo crecerá exponencialmente a medida que aumenten los casos de uso a aplicar y del número de intrusiones a detectar. Además la efectividad de la respuesta se basa simplemente en un tiempo fijo establecido sin tener en cuenta otros factores de contexto que determinen si realmente la repuesta fue eficaz contra la intrusión.

- Además, el uso de modelos matemáticos también se ha aplicado a la construcción de AIRS. Concretamente, en [ZKSY14] se aplica un POMDP (*Partially Observable Competitive Markov Decision Process*) con un juego estocástico de Stackelberg con dos jugadores para decidir la mejor respuesta. Aunque ellos no aplican respuestas proactivas en su sistema.

2.6.3 Predicción de ataques multi-paso basados en HMM

Hay algunos trabajos relacionados con la predicción de ataques multi-paso basados en el uso de Modelos de Markov Ocultos (HMM). En la mayoría de los casos, estos estudios definen los parámetros del HMM de manera similar diferenciándose en gran medida de esta propuesta de tesis (Véase capítulo 6).

La definición del modelo en [HAK07], [HMK08], [SDJC12], [KEAA14] considera los estados de la cadena oculta de Markov como diferentes estados de riesgo del sistema, independientemente de la intrusión que llegue al sistema. Por tanto, estos modelos usan solo un modelo HMM para cualquier tipo de ataque enfocándose únicamente en el nivel de riesgo global de la red de una organización. Por el contrario en [ZGHS09] proponen que los estados ocultos de la cadena estén formados por las fases del ataque. Este último enfoque es similar al considerado para el desarrollo del modelo de predicción propuesto en la tesis doctoral, de esta forma se obtiene un HMM adaptado a cada tipo de ataque multi-paso,

pudiendo predecir el siguiente paso del atacante.

En todos los estudios previos, las observaciones del HMM son recogidas desde alertas de los IDSs. En concreto, las referencias citadas basadas en estados de riesgo utilizan el parámetro de severidad de la alerta o una función de este valor como las observaciones del modelo. Sin embargo, el modelo de predicción propuesto en la tesis doctoral incluye definiciones de vulnerabilidad para incrementar la precisión de los resultados.

A la hora de realizar el entrenamiento del modelo en [SDJC12] no se indica el algoritmo de aprendizaje utilizado en la obtención de las matrices de probabilidad. En [HAK07] y [HMK08] se realiza el entrenamiento solo con datasets simulados en donde no se puede demostrar la veracidad del sistema. En [KEAA14] se basan en el uso de valores aleatorios para el entrenamiento. X. Zan et.al. [ZGHS09] no incluye los valores de las probabilidades a utilizar en las distintas matrices del modelo.

Las validaciones aportadas en los trabajos previos son bastante escasas y solo se enfocan en la representación de la capacidad predictiva sobre la misma secuencia de alertas utilizada en el entrenamiento. Durante el desarrollo de la tesis doctoral se ha llevado a cabo una validación más exhaustiva como se detalla en la sección 6.6.

2.6.4 Predicción de máquinas comprometidas

- Otro punto de vista sobre metodologías predictivas es la aportada por S. Zonlouz et.al. [ZRB+12]. Propone la estimación del estado ciber-físico orientado a seguridad (SCPSE, *security-oriented cyber-physical state estimation*) para identificar las máquinas comprometidas y las medidas de los sensores modificadas maliciosamente. Este trabajo está enfocado sobre los datos alterados maliciosamente sin importar el ataque específico al que está expuesto. SCPSE comienza generando automáticamente un grafo de ataque (AGT) de manera *off-line*. Las transiciones de estados en el grafo recoge todas las posibles rutas que un atacante puede atravesar mediante la explotación de vulnerabilidades de las máquinas específicas que componen la topología de red de la organización. Específicamente, cada transición de estados representa una escalada de privilegios lograda a través de la explotación de una vulnerabilidad. SCPSE analiza los

ataques basado en el grafo de ataque y determina probabilísticamente las máquinas y las medidas de los sensores que han sido comprometidas. Finalmente el AGT es convertido en un HMM para determinar la ruta de máquinas atravesada por el atacante. Una de las desventajas de este modelo es el incremento del tiempo de ejecución a medida que el grafo de ataque aumenta a consecuencia de un mayor número de máquinas en la red de la organización dejando de ser usable en tiempo real en entornos reales.

- H. Du et. al. [DLHY10] propone proyectar los posibles objetivos de ataques futuros en varias etapas a partir de la última máquina atacada. Primero, se evalúa las capacidades de los atacantes basado en los servicios que han sido ya atacados y las vulnerabilidades expuestas del objetivo. Tras ello, se generan Reglas de Inferencia Difusa (*Fuzzy Inference Rules*) a partir de estimaciones de un modelo VLMM (*variable-length Markov Model*) propuesto en [FBY08]. Estas estimaciones se basan en un conjunto de etiquetas XML de alertas reportadas por IDSs, por tanto estas etiquetas dependen del tipo de IDS utilizado. Además, para conseguir estas etiquetas se propone un entrenamiento *off-line* y otro en tiempo real para el árbol de sufijos. Pero en tiempo real el sistema eventualmente se queda sin recursos de memoria si se alimenta indefinidamente con nuevos ataques. En el caso del entrenamiento *off-line* se obtiene un árbol de sufijos limitado a las alertas utilizadas para dicho entrenamiento.

2.6.5 Predicción y correlación de intrusiones basada en otros algoritmos matemáticos

- En [SSDDJ13] utilizan una Máquina de Estados Finita (FSM) para ataques multi-paso en un Sistema de Respuesta a Intrusiones. Específicamente, utilizan un FSM como parte del componente de detección, que envía una alerta al sistema solo cuando se produce un cambio de estado en el FSM sin predecir la evolución del ataque en cada fase. Los autores proponen asignar un peso a cada estado, pero no definen ningún ejemplo para ningún escenario de ataque. Para la mitigación de los ataques multi-paso se basan en la ejecución de un conjunto de respuestas y mide la efectividad de todas ellas mediante una medida de evaluación del riesgo on-line. Esta evaluación determina el

orden en que se lanzaran las respuestas dentro del conjunto en un determinado nivel.

- En [YF07] se propone el uso de HCPN (*Hidden Colored Petri-Net*) para predecir el siguiente objetivo del atacante basado en un modelado matemático. Aunque el propósito del modelo se enfoca en la optimización y correlación de las alertas por parte de los IDSs para reducir el ratio de falsos positivos y negativos. Por otro lado, la solución tiene problemas de rendimiento debido a que las precondiciones y postcondiciones aumentan significativamente con las acciones añadidas al modelo. Estas acciones están basadas en un conjunto de nombres de alertas obtenidas desde IDSs en un escenario de ataque concreto, dependiendo del tipo de IDS usado.

- Una Red Bayesiana Dinámica es propuesta en [FWZZ09] para calcular una probabilidad basada en si una secuencia de llamadas al sistema es o no normal, para determinar el objetivo de dicha secuencia. Este modelo está planteado para su integración dentro de un IDS, con el objetivo de conseguir la detección temprana del tipo de intrusión.

- En [SM13] definen un modelo de predicción como un modelo orientado a objetos derivado desde un grafo de ataque construido tras el reconocimiento de la red. En este trabajo usan conjunto de datos basados en diferentes recursos, como bases de datos de IDSs, la base de datos de vulnerabilidades nacional (NVD, *National Vulnerabilities Database*) y grafos de ataque. El proceso de predicción determina el siguiente paso del atacante y su escenario de ataque, pero el peso de cada estado se asigna manualmente basado en la vulnerabilidad explotada. Además, solo incluye una propuesta teórica con una pequeña prueba de concepto con ataques multi-paso. Por último, si el número de máquinas aumenta, aumentará el número de nodos del grafo, por lo que la propuesta no es escalable.

2.7 Conclusiones

Este capítulo comienza con conceptos básicos sobre ciberseguridad, incluyendo ataques y sistemas de seguridad.

Una de las partes se centra en el estado del arte de algunos sistemas de defensa perimetral, explicando los principales tipos y clasificación de los mismos. La automatización es importante debido a que la mayoría de los ataques, incluidos los APTs, utilizan la automatización. De modo que un método de protección manual no escala contra la automatización de los ataques. Los AIRSs son una tecnología de seguridad que tiene por objetivo seleccionar respuestas contra las intrusiones detectadas por los IDSs de forma automática basado en una serie de métricas para la mitigación de ataques o reducir su impacto.

Los métodos que las organizaciones usan contra los ataques tradicionales se basa en una seguridad reactiva. Esperan un signo visible para tomar una acción de respuesta basado en que el tiempo en el que el atacante causa daño a la organización es mayor que el tiempo que toma la detección y reacción al ataque. Para tratar ataques más sofisticados o ataques basados en APT es necesario moverse desde el enfoque reactivo a la proactividad. Esto se debe a que una vez que se detecta un ataque de este tipo utilizando mecanismos tradicionales de seguridad, es posible que sea tarde y que el atacante haya conseguido su objetivo, por ello la detección temprana es crítica para evitar en la mayor medida los daños en la organización.

El proceso de detección temprana está ligado al concepto de predicción de ataques. La identificación de los distintos pasos de un ataque es imprescindible para llevar a cabo la predicción temprana. De este modo se introduce el concepto de ataque multi-paso como las distintas fases que realiza un atacante para conseguir un objetivo. Además, se muestran varios ejemplos de escenarios de ataque propuestos en el entorno de investigación en este ámbito.

Otro parte clave para la evolución de los sistemas de protección es hacer frente al dinamismo de los ataques con el concepto de auto-aprendizaje. En este sentido, si los sistemas de protección son capaces de adaptarse de manera autónoma, mejorará la eficacia contra la sofisticación de los ataques. A este respecto hay algunos estudios previos que basan el aprendizaje de los AIRSs a partir de la eficiencia en respuestas anteriores como es el caso de ADEPTS con una solución poco rigurosa y la propuesta incluida en [ML13] con mayor rigor pero con el inconveniente de tener que asignar un número de umbrales muy alto.

Tras el análisis llevado a cabo, este trabajo de tesis tiene por objetivo generar un modelo

predictivo y un mecanismo para el aprendizaje autónomo de los sistemas de respuestas a intrusiones de tal manera que se obtenga un *Sistemas de Respuestas a Intrusiones Proactivo y Autónomo,* aunque estos conceptos también puedan ser utilizado en otro tipo de sistemas, como por ejemplo para el análisis dinámico del riesgo.

3

MECANISMOS DE APRENDIZAJE AUTOMÁTICO

3.1 Introducción

Ambas contribuciones de la tesis doctoral se basan en la aplicación de un mecanismo de Aprendizaje Automático. De este modo es posible dotar al AIRS de auto-aprendizaje y de capacidad de predicción frente a los distintos ataques.

El Aprendizaje Automático o *Machine Learning* forma parte de una rama de la Inteligencia Artificial cuyo objetivo es desarrollar técnicas de aprendizaje. Dentro de este grupo, hay diversas técnicas de aprendizaje [HA03]. Se trata de algoritmos que son capaces de generalizar comportamientos a partir de muestras basadas en observaciones pasadas. El Aprendizaje Automático es una disciplina científica que se focaliza en la problemática de hacer predicciones precisas de un conjunto de datos basado en observaciones pasadas.

Hay una gran variedad de métodos de Aprendizaje Automático, cada uno de los cuales tienen distintas propiedades. En la Tabla 3.1.1 muestra una comparativa de varios modelos de

Sistema	Robustez	Solución única	Transparencia	Eficiencia	Tolerancia a ruidos	Sobreajuste	Datos incompletos	Entrenamiento	Test	Parámetros adicionales
Máxima entropía	+	+	-	+	+	-	+	lento	rápido	+
Boosting	+	+	+	+	-	+	+	lento	rápido	-
HMM	+	+	+	+	-	-	+	lento	lento	-
Red Neuronal	+	-	-	+	+	+/-	-	lento	lento	-
Red Bayesiana	+/-	+	+	-	+	+	+	lento	lento	+
Clasificador de Naive Bayes	+/-	+	+	-	+	-	+	lento	lento	-
SVM	+/-	+	-	+	-	-	-	lento	lento	-

Tabla 3.1.1: Comparativa de sistemas de Aprendizaje Automático

aprendizaje automático ampliamente utilizados dentro de distintos dominios. Para mayor información sobre cada uno de estos modelos consulte la referencia [Mac10]. En concreto se puede observar características básicas de cada uno de ellos, como la robustez del modelo, la transparencia, la eficiencia, la tolerancia a ruidos, como maneja los sobreajustes y los datos incompletos, el tiempo de entrenamiento y test.

Hay una gran variedad de sistemas de aprendizaje que se utilizan hoy en día para todo tipo de aplicaciones. La clasificación de estos algoritmos se divide en: supervisados, no supervisados, semi-supervisados, reforzado, por transducción, y el denominado learning to learn.

- Los algoritmos supervisados producen una función de correspondencia tras mostrarle las entradas y las salidas deseadas del sistema; con ello se consigue la supervisión del aprendizaje por un maestro, es decir, que el aprendizaje va guiado por el conocimiento previo aportado.

- En los algoritmos no supervisados todo el proceso de modelado se lleva a cabo sobre

un conjunto de ejemplos formado tan sólo por entradas al sistema. No se tiene información sobre las categorías de esos ejemplos. Por lo tanto, en este caso, el sistema tiene que ser capaz de reconocer patrones para poder etiquetar las nuevas entradas.

- Los semi-supervisados combinan los dos algoritmos anteriores para poder clasificar de manera adecuada.

- El aprendizaje reforzado observa el mundo que le rodea, es decir, la información de entrada es la retroalimentación que obtiene como respuesta a sus acciones. Por lo tanto, el sistema aprende a base de ensayo-error.

- El aprendizaje por transducción es similar al aprendizaje supervisado, pero no construye de forma explícita una función. Trata de predecir las categorías de los futuros ejemplos basándose en los ejemplos de entrada, sus respectivas categorías y los ejemplos nuevos del sistema.

- Learning to learn el algoritmo aprende de su propio bias basado en experiencias previas.

Los sistemas de aprendizaje tienen en común una serie de desventajas como la necesidad de una gran cantidad de ejemplares para el entrenamiento, fases de entrenamiento lentas y necesidad de proponer métodos para evitar sobreajuste y convergencia en máximos/mínimos locales.

A continuación se van a detallar dos métodos de Aprendizaje Automático, que van a ser utilizados para el desarrollo de las contribuciones de la tesis doctoral. El primero de ellos, las Redes Neuronales Artificiales, se clasifican dentro de los algoritmos bio-inspirados y el segundo, los Modelos de Markov Ocultos, se clasifican en la categoría de aprendizaje estadístico.

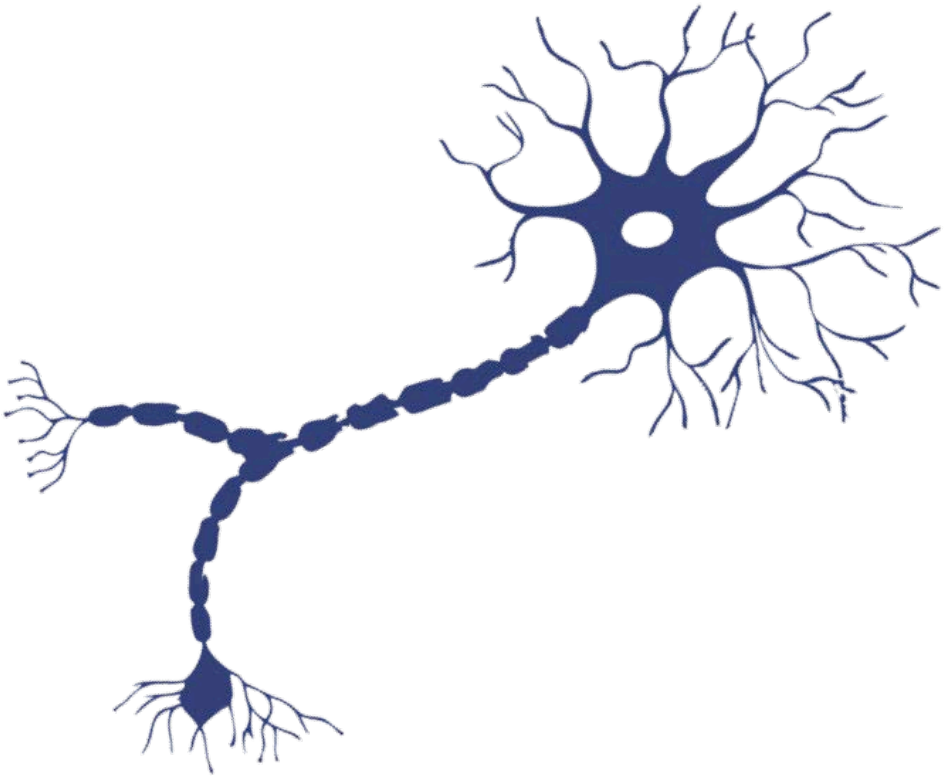


Figura 3.2.1: Neurona del sistema nervioso [MM17]

3.2 Redes Neuronales Artificiales

Las Redes Neuronales Artificiales son redes interconectadas masivamente en paralelo, de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interaccionar con los objetos del mundo real del mismo modo que hace el sistema nervioso [Koh88]. Las neuronas (Figura 3.2.1) están interconectadas a través de su sinápsis, permitiendo así la transmisión de información [Heao8].

La Figura 3.2.2 representa una ejemplo de Red Neuronal Artificial. Dicha red está formada como mínimo por dos unidades de procesamiento que tienen activación y peso. Las conexiones entre neuronas no son todas iguales, por lo que es necesario asignar un peso a cada una de las conexiones, W. Los pesos obtenidos tras la fase de entrenamiento determinan la salida de la red, por lo que se podría decir que son la memoria de la Red Neuronal.

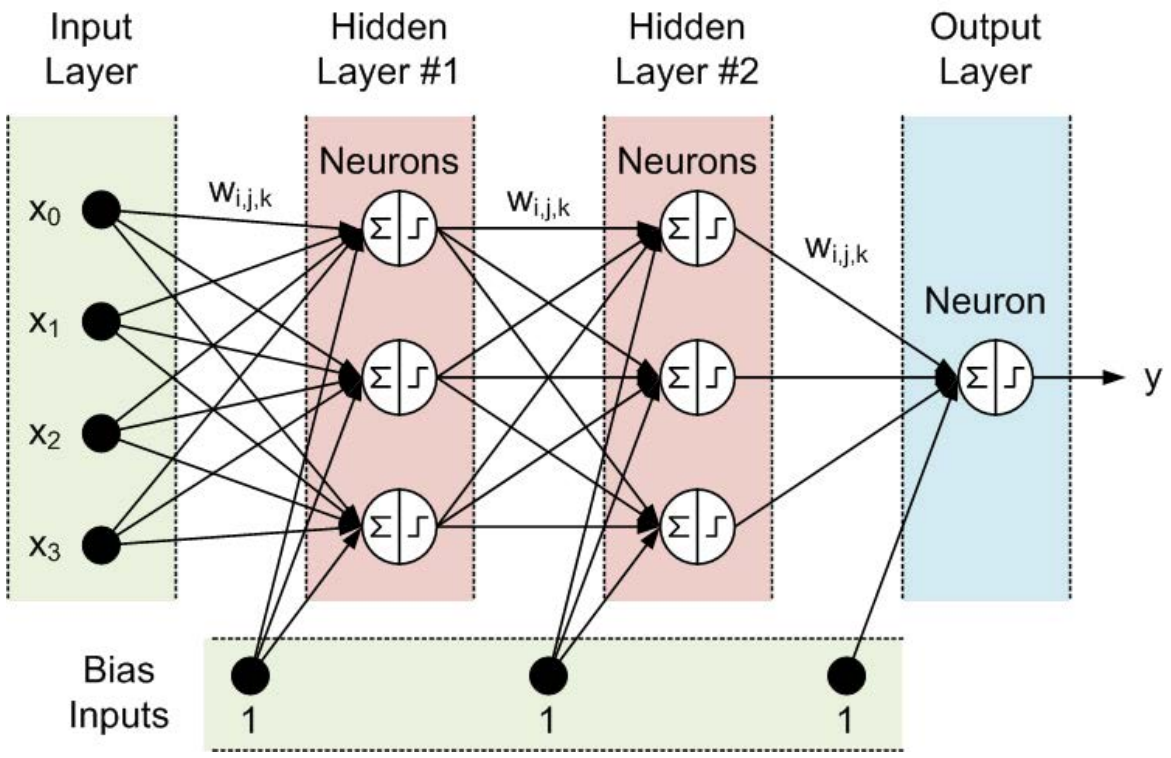


Figura 3.2.2: Ejemplo de red Neuronal Artificial [Com16]

A continuación se indican la ventajas del uso de Redes Neuronales:

- Aprendizaje adaptativo: aprende mediante entrenamiento.
- Autoorganización: propia representación de la información.
- Tolerancia a fallos
- Robustez frente a destrucción parcial.
- Operación en tiempo real: Facilidad de realización HW.
- Integración modular: fácil inserción dentro de la tecnología existente.
- Son una eficiente herramienta de clasificación que puede ser aplicado a problemas complejos con gran cantidad de parámetros.
- Es confiable en la predicción de problemas de regresión y clasificación.

Como desventajas se encuentran:

- El problema de las Redes Neuronales es 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.
- Hay que ajustar cuidadosamente un gran número de parámetros para obtener buenos resultados.
- La fiabilidad de las predicciones se degrada con la presencia de múltiples soluciones porque tiene muchos mínimos locales.
- Es difícil determinar y entender su rendimiento.
- Son muy lentas tanto en fase de entrenamiento como de validación
- Sensibles a los conjuntos de datos incompletos
- Problema de sobreajuste, pero con la utilización de validación cruzada se puede evitar obteniendo un buena clasificación en entornos ruidosos.

3.2.1 Problemas no adecuados para resolver con Redes Neuronales Artificiales

- Problemas que se pueden escribir fácilmente como diagrama de flujo. Si el problema puede ser definido en pasos bien definidos con una técnica normal de programación será suficiente.
- Problemas que no sean cambiantes: las Redes Neuronales se caracterizan por su habilidad para aprender, en el caso de que el problema no sea cambiante, no hay razones para aplicar estos algoritmos. Incluso si lo intentas implementar, empieza a divergir, produciendo resultados inesperados.
- Problemas en los que se sabe exactamente como se derivó la solución: Las Redes Neuronales son útiles para solventar problemas para las cuales han sido entrenadas, pero no pueden explicar sus razones.

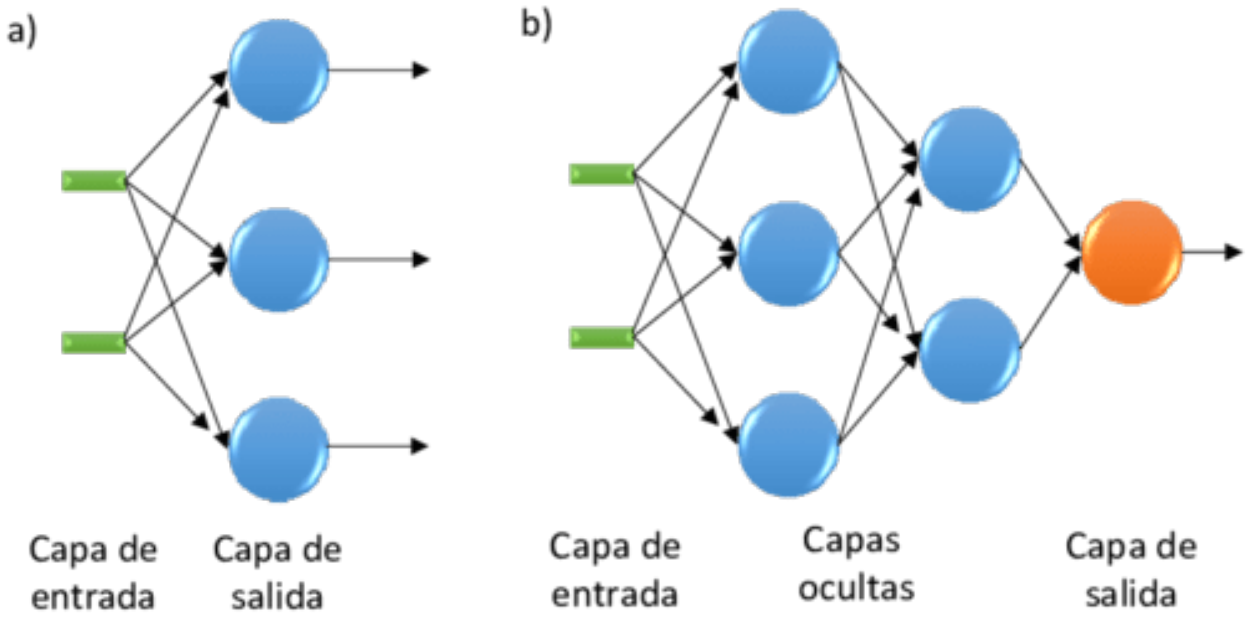
3.2.2 Problemas adecuados para resolver con Redes Neuronales Artificiales

Las Redes Neuronales pueden solventar un problema con menos líneas de código que un algoritmo de programa tradicional. 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. El reconocimiento de patrones es el uso más común de las Redes Neuronales. Para este tipo de problemas se le presenta a la Red Neuronal un patrón, ya sea imagen, sonido o cualquier otro dato. Entonces la Red Neuronal intenta determinar si el dato de entrada concuerda con un patrón que ha sido entrenado para reconocer. 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.

3.2.3 Topología de las Redes Neuronales Artificiales

En la organización de las Redes Neuronales no es habitual que neuronas del mismo nivel se conecten entre sí. Frecuentemente las conexiones entre niveles son totales. Hay dos posibles clasificaciones de Redes Neuronales:

1. Por estructura (véase Figura 3.2.3):
    - Monocapa: con una sola unidad de procesamiento o varias pero en la misma capa.
    - Multicapa: Varias unidades de procesamiento en varias capas intermedias o capas ocultas.
2. Por el flujo que sigue la información:
    - Flujo directo: La información va en una sola dirección (Feed Forward)
    - Flujo recurrente o asociativo: La información puede ir tanto hacia delante como hacia atrás. Se dividen en autoasociativas y heteroasociativas.



(a) Monocapa **(b)** Multicapa

Figura 3.2.3: Red Neuronal Artificial [MM14]

Una representación de una topología genérica de Red Neuronal Artificial se representa en la Figura 3.2.2. Toda Red Neuronal Artificial está compuesta por la capa de entrada y una capa de salida, con una o más neuronas por capa. En la Figura 3.2.2 se puede ver que la capa de entrada está compuesta por cinco neuronas, teniendo en cuenta la neurona de *Bias* y por una neurona en la capa de salida.

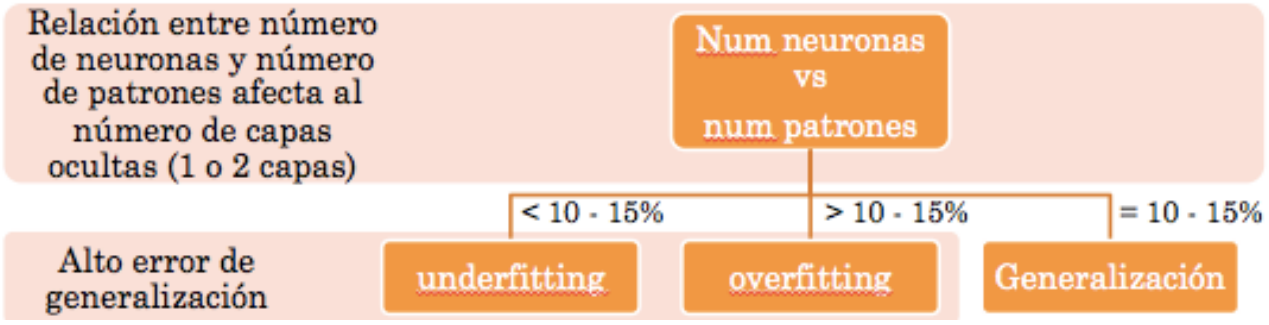


Figura 3.2.4: Esquema de relación entre el número de parámetros y el número de patrones

3.2.3.1 Relación entre el número de parámetros y el número de patrones

A la hora de diseñar la estructura de la Red Neuronal hay que tener en cuenta que se debe disponer de un número suficiente de muestras para que haya generalización de los resultados.

Se considera buena una relación entre un 10-15 % siguiendo la ecuación (3.1) y el esquema presentado en la figura Figura 3.2.4.

$$\#parmetros < \frac{[10-15]}{100} \cdot \#patrones \tag{3.1}$$

Cuanto menor sea el número obtenido, mayor capacidad de generalización.

También hay unas reglas llamadas *Rules of thumb* que ayudan a determinar el número de neuronas más apropiado, algunas de ellas son:

- Para la selección del número de neuronas ocultas se utiliza la regla general, $h = (\frac{2}{3}) \cdot (n+m)$, donde n es el número de neuronas de entrada y m es el número de neuronas de la salida. Esto quiere decir que, habitualmente el número de neuronas en la capa oculta es $\frac{2}{3}$ del número total de neuronas de las capas de entrada y salida.
- El número de neuronas de la capa oculta nunca requerirá ser mayor que dos veces el número de neuronas de la entrada

Estas reglas deben ser tenidas en cuenta a la hora de diseñar la topología de una Red Neuronal y seleccionar el número de neuronas adecuado en las capas ocultas si las hubiere.

- 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).

El número de capas ocultas debe ser tenido en cuenta ya que cuantas más capas ocultas el aprendizaje será más rápido al principio, pero pruede incrementar en el número de pará-

metros utilizados y por tanto en el número de patrones de entrenamiento. Es por ello que añadir más capas no supone una gran mejora. La mayoría de los problemas se resuelven con 1 o 2 capas ocultas.

3.2.4 Función de activación o transferencia

Una Red Neuronal ya entrenada, toma muestras de entrada y las clasifica en grupos. Estos grupos pueden ser difusos, es decir, los límites no están claramente definidos, o con bordes definidos si se seleccionan umbrales.

La función de activación, también también llamada función de transfrencia, define un umbral fijo o una función umbral para normalizar tras el procesamiento de la información, limitando la amplitud de la salida de una neurona. Algunos de las funciones de activación más comúnmente usadas se utilizan para solventar problemas no-lineares [KO11]. Estas funciones son:

- Función sigmoide uni-polar: Está función es especialmente ventajosa para su uso en Redes Neuronales entrenadas con el algoritmo de BackRetropropagaciónpropagation porque minimiza la capacidad de cálculo del entrenamiento. Hace corresponder las salidas de las neuronas en el intervalo [0-1].

$$g(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

- Función sigmoide bipolar: Esta función de activación es igual que la anterior, pero se aplica en el rango de salida [-1,1].

$$g(x) = \frac{2}{1 + e^{-x}} - 1 \tag{3.3}$$

- Función tangente hiperbólica: Esta función es fácilmente definida como el ratio entre el seno y el coseno hiperbólico. El rango de valores de salida se encuentran entre [-1-1].

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.4}$$

- Función de base radial (RBF): Está basada en una Curva Gausiana. Toma un parámetro que determina el centro de la función usado como valor deseado. Es una función de valor real cuyo valor depende de la distancia desde el origen, $g(x) = g(||x||)$ o alternativamente sobre la distancia desde otro punto c, llamado centro, $g(x, c) = g(||x - c||)$. Las sumas de las funciones de base radial se usan generalmente para aproximar a funciones determinadas. Este proceso de aproximación puede interpretarse como un tipo simple de Red Neuronal con una sola capa. RBF se utiliza generalmente para construir funciones de la forma:

$$y(x) = \sum_{i=1}^{N} w_i \cdot g(||x - c_i||) \tag{3.5}$$

- Función de sección cónica (CSF): Está basada en una sección de un cono que toma un parámetro para determinar el valor de ángulo de la función.

$$f(x) = \sum_{i=1}^{N+1} (a_i - c_i) \cdot w_i - cosw_i \cdot (||a - c_i||) \tag{3.6}$$

Donde a_i es un coeficiente de entrada, c_i es el centro y w_i es el peso de cada una de las neuronas.

3.2.5 Entrenamiento de las Redes Neuronales Artificiales

Las neuronas que conforman una Red Neuronal están interconectadas a través de su sinápsis, 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. Lo más habitual, es que todas las neuronas estén interconectadas en la topología y en caso de que no haya conexión entre dos neuronas, se indica poniendo el peso de conexión a cero. Estos pesos son los que determinan la salida de la red, por lo que se puede decir que conforman la memoria de la Red Neuronal.

El entrenamiento es el proceso por el que se asignan estos pesos de conexiones. La mayoría de los algoritmos de entrenamiento empiezan asignándoles un número aleatorio a la matriz

de pesos. Entonces, tras cada una de las validaciones los pesos son modificados en base al rendimiento de la Red Neuronal y a la validez de los resultados. Este proceso es repetido hasta que el error de validación está dentro de un límite aceptable.

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

- Supervisado: Se sabe a priori a que salida se corresponde cada uno de los datos de entrada, es decir, se sabe cómo son las características a clasificar. El hecho de conocer la salida implica que el entrenamiento se beneficia de la supervisión de un maestro. Necesitan un conjunto de datos de entrada previamente clasificado o cuya respuesta objetivo se conoce. Es el tipo de entrenamiento más habitual. La red supervisada realiza una serie de iteraciones o épocas hasta que la salida se corresponde con la salida esperada con un ratio de error razonablemente bajo o 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. Ejemplos de este tipo de redes son: el perceptrón simple, la red Adaline, el perceptrón multicapa, Retropropagación, y la memoria asociativa bidireccional.

- No supervisado: No se sabe que categorías se tienen o no se sabe las características a clasificar, entonces se hace una estadística para determinar el patrón. Por ello, se suele utilizar para clasificar las entradas en varios grupos. Al igual que en el caso de aprendizaje supervisado, se ejecutan muchas épocas. A medida que progresa el entrenamiento, los grupos de clasificación son descubiertos por la Red Neuronal. Ejemplos 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 Kohonen, mapas autoorganizados y las redes de resonancia adaptativa (ART).

Hay varios métodos híbridos que combinan aspectos de ambos tipos de entrenamiento supervisado y no supervisado. Entre ellos se encuentran:

- Redes híbridas: son un enfoque mixto en el que se utiliza una función de mejora para facilitar la convergencia. Un ejemplo de este último tipo son las redes de base radial.

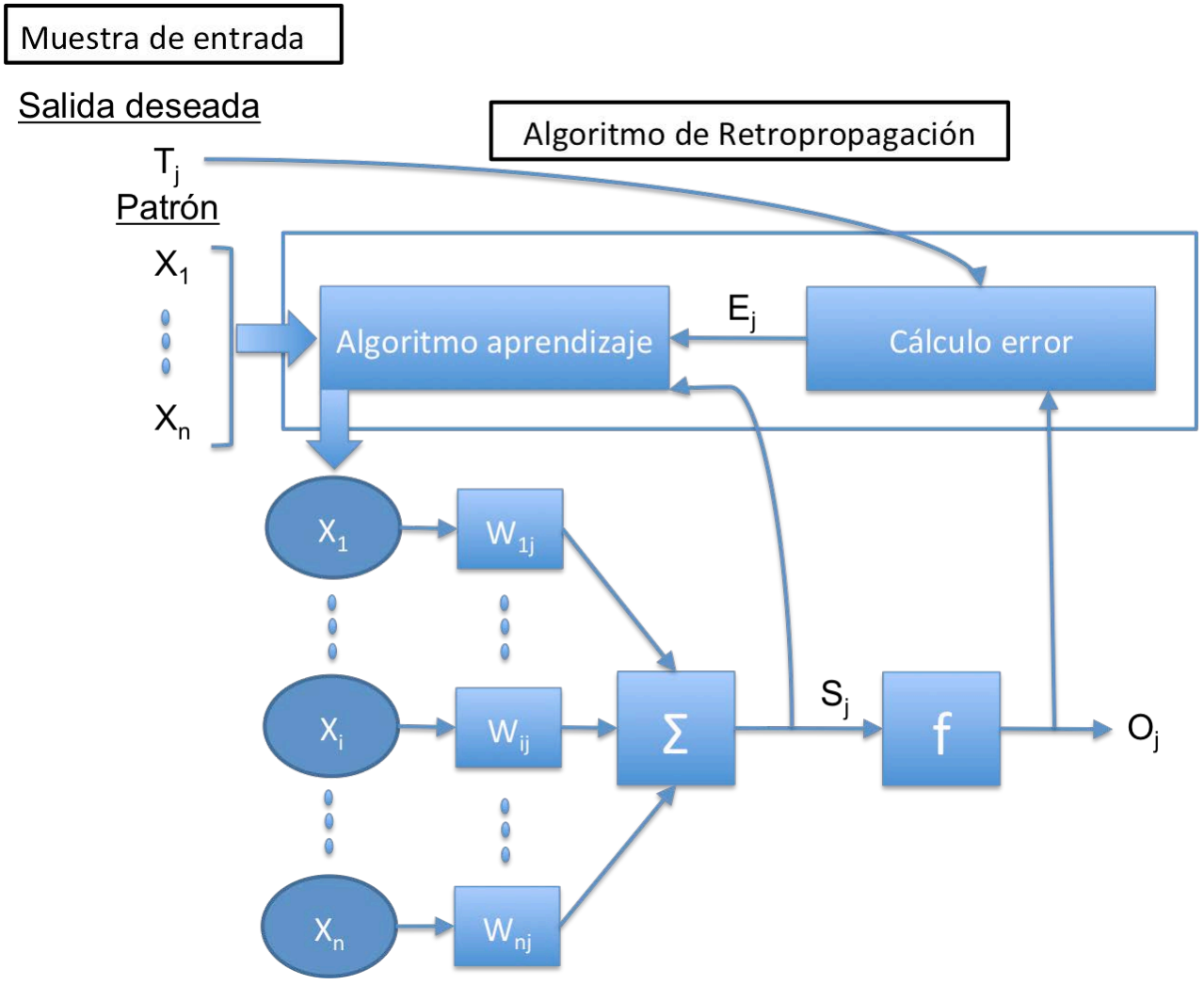


Figura 3.2.5: Esquema de funcionamiento del algoritmo de Retropropagación

- Aprendizaje reforzado: se sitúa a medio camino entre el supervisado y el autoorganizado. Se proporciona datos de entrada a la Red Neuronal, pero sin indicar la salida. Sin emabrgo para cada salida la Red Neuronal dice si la salida es correcta o incorrecta dada la entrada.

3.2.5.1 Algoritmo de Retropropagación

El algoritmo de Retropropagación (*Backpropagation Algorithm*) es uno de los algoritmos supervisados más utilizados en todos los ámbitos de aplicación y su funcionamiento se muestra en el esquema de la Figura 3.2.5:

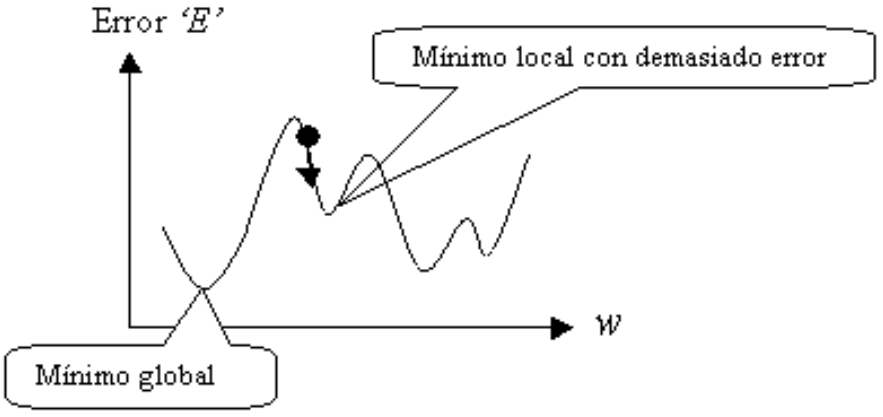


Figura 3.2.6: Ejemplo de la gráfica de la función de error [Her05]

1. En cada iteración, el algoritmo va incluyendo una a una las distintas muestras disponibles en el fichero de entrenamiento como entrada en las variables X y el valor deseado, T_j.

2. El algoritmo de retropropagación está basado en el método de descenso del gradiente para aproximarse al mínimo error cuadrático medio, aplicando las fórmulas (3.7) y (3.8) en cada uno de los pesos de las neuronas de la Red Neuronal, donde δ_j es el error propagado, x_i es el valor de entrada y a es el factor de aprendizaje.

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} \tag{3.7}$$

$$\Delta w_{ij} = a \cdot \delta_j \cdot x_i \tag{3.8}$$

3. Una vez aplicada a la entrada los nuevos pesos calculados y junto con la función de activación se obtiene el valor de salida, O_j.

4. Este valor obtenido será comparado con la salida deseada T_j, calculando el error que se produce entre ellas.

Una de las desventajas de este algoritmo es que la función de error tiene normalmente muchos mínimos locales donde el algoritmo puede tender a converger (véase Figura 3.2.6).

Los pesos sinápticos pueden depender del gradiente medio de los puntos ambientales, en lugar de depender de un solo punto, para evitar quedar atrapados en un mínimo local. Pero esta modificación requiere un gran esfuerzo computacional y no sería eficiente. Es por ello que hay dos alternativas para mejorar el algoritmo y prevenir frente a mínimos locales: El factor de aprendizaje adaptativo y la retropropagación con momento [RHW85].

Factor de aprendizaje adaptativo El factor de aprendizaje adaptativo se basa en la modificación del factor de aprendizaje inicial, $a(t)$, adaptándose según las necesidades del aprendizaje:

- Si las pendientes tienen el mismo signo, los pesos cambiarán más rápido basado en un valor de capa k, como por ejemplo 0,035:

$$a(t+1) = a(t) + k \tag{3.9}$$

- Si las pendientes tienen distinto signo, los pesos cambiarán más lentamente como se muestra en la ecuación (3.10) con un valor por ejemplo de $\gamma = 0,3333$

$$a(t+1) = a(t) + (1-\gamma) \tag{3.10}$$

Aprendizaje con momentos El aprendizaje por momentos tiene en cuenta el gradiente de la iteración anterior para realizar un promedio con el gradiente de la iteración actual, como se muestra en la ecuación (3.11), por ejemplo con el valor de $\mu = 0,9$. 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. 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.

$$\Delta w_{ij} = a \cdot \delta_j \cdot x_i + \mu \cdot (w_{ij}(t) - w_{ij}(t-1)) \tag{3.11}$$

3.2.6 Validación

Esta última etapa es muy importante porque permite determinar si se requiere entrenamiento adicional. Para ello se debe tener otro conjunto distinto al de entrenamiento para evitar sobreajustes. El término sobreajuste se refiere a la no generalización de la clasificación llevada a cabo por el algoritmo. Uno de los métodos más utilizados para evitar sobreajustes en la etapa de validación de la Red Neuronal Artificial es la denominada *validación cruzada*, que consiste en dividir la muestra en dos partes con ejemplos de todas las clases:

- Conjunto de entrenamiento para construir el modelo
- Conjunto de test para validar la generalización del modelo

Otro factor a tener en cuenta para conseguir un buen entrenamiento de la Red Neuronal es que los patrones del conjunto de entrenamiento deben ser suficientemente dispares y de diversas clases para evitar el sobreentrenamiento de la red.

3.3 Modelos de Markov Ocultos

Un Modelo de Markov Oculto (HMM) está compuesto por dos procesos estocásticos [Rab89], de tal manera que un proceso estocástico subyacente que no es observable, es decir, que está oculto, puede ser observado a través de otro conjunto de procesos estocásticos que produce una secuencia de observaciones. En la Figura 3.3.1 se puede ver la representación gráfica:

1. El proceso oculto está representado por la variable aleatoria x(t) y se corresponde con los estados de la cadena de Markov.
2. El proceso de observaciones viene representado por la variable aleatoria y(t).

Los HMM describen en términos probabilísticos transiciones de estados a través de eventos que suceden de acuerdo a una distribución de probabilidad, es decir, probabilidades de transición de estados y probabilidades de observación. Se considera una variante determinista de las máquinas de estados finitos y una simplificación de una Red Bayesiana Dinámica.

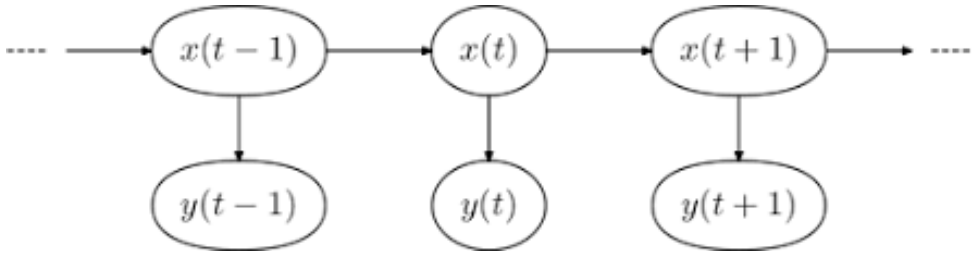


Figura 3.3.1: Procesos estocásticos de un HMM [Wik17]

El aprendizaje se basa en encontrar, dada una secuencia de observaciones o un conjunto de tales secuencias, el mejor conjunto de probabilidades de transición de estado y de probabilidades de observación.

Como ventajas de este tipo de modelos se encuentra:

- Análisis matemático y teórico de los resultados y procesos
- Modularidad, es decir, facilidad de introducir nuevos datos sin afectar al aprendizaje anterior
- Transparencia del modelo por lo que es entendible y por tanto puede ser fácilmente analizado y mejorado si es necesario.
- En el caso supervisado usa conocimiento previo para limitar el proceso de entrenamiento.
- El Modelo de Markov Oculto es un modelo de clasificación efectiva que puede ser aplicado a tareas complejas
- El modelo es robusto y puede ser combinado con pequeñas HMMs
- La falta de datos no afecta a la precisión de la clasificación

Por otro lodo, como desventajas de los HMM:

- No se puede evitar el sobreajuste por lo que es sensible a datos ruidosos.
- El número de parámetros a establecer es enorme: Por ejemplo para 3 estados se evalúan 15 parámetros.

- Requiere una gran cantidad de datos para un buen entrenamiento, sobre todo en caso de entrenamiento no supervisado
- Lentos en fase de entrenamiento como la mayoría de sistemas de aprendizaje automático.
- Supone estados independientes. Aunque esto normalmente no es cierto para su implementación en casos reales.

Formalmente, un HMM discreto de primer orden es definido por $\lambda = (\Sigma, S, A, Q, \pi)$. Estos parámetros son explicados a continuación:

1. Σ representa todas las posibles observaciones del modelo.

$$\Sigma = \{v_1, v_2, \ldots, v_M\} \tag{3.12}$$

2. S es el conjunto de estados de la cadena de Markov.

$$S = \{s_1, s_2, \ldots, s_N\} \tag{3.13}$$

3. A es la matriz de probabilidad de transiciones que describe las transiciones entre estados del conjunto S. La ecuación (3.14) representa esta matriz, donde i y j representan un estado del conjunto S.

$$A = \{\{a_{ij}\}_{NxN} | a_{ij} = P(x_t = j | x_{t-1} = i)\} \tag{3.14}$$

4. Q es la matriz de probabilidad de observación que se compone de un vector de probabilidad para cada estado (definido como q_i). En la ecuación (3.15), i representa un estado del conjunto S y j se corresponde con una observación del conjunto Σ.

$$Q = \{\{q_{ij}\}_{NxM} | q_i = \{q_1, q_2, \ldots, q_M\}\} \tag{3.15}$$

5. π es el vector de distribución inicial, el cual indica la probabilidad de que cada uno de los estados sea el estado inicial.

$$\pi = \{\pi_1, \pi_2, \ldots, \pi_N\} \tag{3.16}$$

Las observaciones obtenidas a cada instante se representan en la secuencia de observaciones siguiendo la siguiente expresión: $O = \{o_1, o_2, \ldots, o_T\}$, donde cada componente o_t de la secuencia es un símbolo de Σ. Hay tres problemas básicos relacionados con los HMMs [Rab89]:

Problema 1 Calcular eficientemente la probabilidad de la secuencia de observación, $P(O|\lambda)$, usando el algoritmo Forward-Backward [YK03].

Problema 2 Encontrar la ruta más probable de estados dado el modelo y la secuencia de observación, usando el algoritmo de Viterbi.

Problema 3 Seleccionar los parámetros A, Q, y π para maximizar la $P(O|\lambda)$.

En los siguientes subapartados se explicarán distintos algoritmos para afrontar estos problemas.

3.3.1 Probabilidad de observación de la secuencia de estados

El Problema 1 de los HMM se refiere al cálculo de la probabilidad de la secuencia de observación $O = \{o_1, o_2, \ldots, o_T\}$ dado el modelo λ, es decir, la $P(O|\lambda)$. El modo más sencillo de hacerlo es enumerando todas las posibles secuencias de estados de longitud T. Considerando una secuencia de estados fijo, $X = \{x_1, x_2, \ldots, x_T\}$, la probabilidad de la secuencia de observación es:

$$P(O|X, \lambda) = \Pi_{t=1}^{N} P(o_t|x_t, \lambda) \tag{3.17}$$

Asumiendo la independencia estadística entre observaciones:

$$P(O|X,\lambda) = q_{x_1}(o_1) \cdot q_{x_2}(o_2) \ldots q_{x_T}(o_T) \tag{3.18}$$

La probabilidad de dicha secuencia de estados puede ser escrita como:

$$P(X|\lambda) = \pi_{x_1} \cdot a_{x_1 x_2} \cdot a_{x_2 x_3} \ldots a_{x_{T-1} x_T} \tag{3.19}$$

La probabilidad conjunta de que ocurra O y X simultáneamente es el producto de las dos ecuaciones mencionadas:

$$P(O,X|\lambda) = P(O|X,\lambda) \cdot P(X|\lambda) \tag{3.20}$$

Por tanto, la probabilidad de la secuencia de observación dado el modelo λ, se obtiene con la suma de la probabilidad conjunta para cualquier secuencia de estados posible:

$$P(O|\lambda) = \Sigma_{todoX} P(O|X,\lambda) \cdot P(X|\lambda) \tag{3.21}$$

De acuerdo a esta ecuación, la $P(O|\lambda)$ involucra del orden de $2T \cdot N^T$ cálculos debido a que para cada instante t hay N estados posibles (N^T posibles secuencias de estados) y para cada secuencia de estados los 2T cálculos son debidos al sumatorio de la ecuación (3.21). Este cálculo es inviable, incluso para cadenas de Makov pequeñas [Rab89].

Algoritmo Forward-Backward El algoritmo Forward-Backward es un procedimiento más eficiente computacionalmente para el cálculo de la probabilidad de la secuencia de observación, $P(O|\lambda)$. Se considera la variable *forward*, $a_t(i)$, definida como:

$$a_t(i) = P(\{O_1, O_2, \ldots, O_t\}, x_t = s_i|\lambda) \tag{3.22}$$

De manera similar, se considera la variable *backward*, $\beta_t(i)$, definida como:

$$\beta_t(i) = P(\{O_{t+1}, O_{t+2}, \ldots, O_T\}, x_t = s_i|\lambda) \tag{3.23}$$

El procedimiento inductivo para obtener cada una de estas variables es mostrada en [Rab89] y requiere del orden de $N^2 \cdot T$ cálculos para cada una de ellas. Estas variables son utilizadas para ayudar a resolver tanto el Problema 2 como el Problema 3 de los HMM.

3.3.2 Secuencia de estados

Uno de los problemas básicos de los HMM es encontrar la ruta más probable dada una secuencia de observación y un modelo λ (Problema 2). El estado más probable para cada observación está basado en la probabilidad condicional (3.24):

$$\gamma_t(i) = P(x_t = s_i|O, \lambda) \tag{3.24}$$

$$x_t = \max_{1 \leq i \leq N}[\gamma_t(i)], 1 \leq t \leq T \tag{3.25}$$

Sin embargo, si el modelo HMM tiene transiciones de estado con probabilidad cero, esta secuencia de estados no sería correcta porque (3.25) determina simplemente el estado más probable en cualquier momento, sin considerar la probabilidad de que la secuencia de estados pueda ocurrir. Como solución se aplica el algoritmo de Viterbi para encontrar la secuencia de estados más probable.

Algoritmo de Viterbi El algoritmo de Viterbi encuentra la mejor secuencia de estados, $X = \{x_1, x_2, \ldots, x_T\}$, para una secuencia de observaciuón dada ($O = \{o_1, o_2, \ldots, o_T\}$). La mejor puntuación a lo largo de una única trayectoria en el tiempo t se define por $\delta_t(i)$, el cual considera las primeras t observaciones y finaliza en el estado s_i:

$$\delta_t(i) = \max_{x_1,\ldots,x_{t-1}} P(x_1 \ldots x_t = s_i, o_1 \ldots o_t|\lambda) \tag{3.26}$$

La clave del algoritmo de Viterbi es que si se conoce $\delta_t(i)$, por inducción:

$$\delta_{t+1}(i) = [\max_i \delta_t(i) \cdot a_{ij}] \cdot q_j(o_{t+1}) \tag{3.27}$$

Finalmente, el algoritmo calcula la mejor secuencia de estados con una complejidad de $O(T \cdot N^2)$. En [For73], se explica este algoritmo con mayor detalle.

3.3.3 Algoritmos de entrenamiento

Uno de los problemas más importantes que envuelven los HMMs es el ajuste de los parámetros A, Q, y π para maximizar la probabilidad de la secuencia de observación dado el modelo λ (Problema 3). No hay formas analíticas para seleccionar los valores de estas probabilidades, pero es posible calcular los valores de probabilidad a partir de un entrenamiento supervisado y no supervisado.

3.3.3.1 Entrenamiento no supervisado

Existen diferentes algoritmos de entrenamiento no supervisados que pueden usarse para calcular dichas probabilidades como son los algoritmos de Baum-Welch, EM (Expectation Maximisation), GEM (Generalised EM), y el Método de descenso del gradiente. Dentro de todos estos, Baum-Welch es el algoritmo más usado para el entrenamiento de este tipo de modelos, debido a que es un caso particular de EM aplicado al entrenamiento de HMMs.

Algoritmo Baum-Welch Formalmente el algoritmo Baum-Welch define la función auxiliar de Baum:

$$S(\lambda_k, \bar{\lambda}) = \sum_{Q} P(S|O, \lambda) \cdot log[P(O, S|\bar{\lambda})] \tag{3.28}$$

Esta función depende de los parámetros previos del modelo (λ) y de los nuevos parámetos $(\bar{\lambda})$ obtenidos tras la aplicación del algoritmo. En orden a describir el procedimiento de reestimación de parámetros, es decir, la modificación iterativa de los parámetros del HMM, se define:

$$\gamma_t(i) = P(x_t = s_i|O, \lambda) \tag{3.29}$$

$$\xi_t(i,j) = P(x_t = s_i, x_{(t+1)} = s_j | O, \lambda) \tag{3.30}$$

$\gamma_t(i)$ and $\xi_t(i,j)$ son calculados a partir de las definiciones de las variables *forward* y *backward* [Rab89]. A partir de $\gamma_t(i)$ se puede obtener el número de transiciones realizadas desde s_i y el número esperado de veces en el estado i. Además, se puede calcular el número esperado de transiciones desde s_i a s_j de manera similar.

Usando las fórmulas anteriores, se puede crear un método para reestimar los parámetros del HMM. Un conjunto de fórmulas de reestimación razonables para A y Q son:

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{3.31}$$

$$\overline{q_j}(h) = \frac{\sum_{t=1, o_t=v_h}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{3.32}$$

Basado en este procedimiento, si iterativamente se usa $\overline{\lambda}$ en vez de λ y se repite el cálculo llevado a cabo para la reestimación, es posible mejorar la probabilidad de que el modelo concuerde con la observación O. El resultado final de este procedimiento de reevaluación se denomina estimación de máxima verosimilitud del HMM. La teoría de máxima expectación indica que maximizando la función (3.28) conduce a un incremento de la probabilidad:

$$\max_{\overline{\lambda}} [S(\lambda, \overline{\lambda})] \Rightarrow P(O|\overline{\lambda}) \geq P(O|\lambda) \tag{3.33}$$

La función de probabilidad converge a un punto crítico. Debe tenerse en cuenta que el algoritmo forward-backward solo conduce a un máximo local. Las fórmulas reestimadas, (3.31) y (3.32), pueden ser derivadas directamente maximizando la función auxiliar de Baum usando using técnicas estándar de optimización restringida (Lagrange multipliers). Además, esto puede ser interpretado como una implementación del algoritmo EM [DLR77], en el cual el paso de expectación es el cálculo de la función auxiliar de Baum, y la fase de modificación es la maximización. Así, las ecuaciones de reestimación del algoritmo Baum-Welch son esen-

cialmente idénticas a los pasos del algoritmo EM para el problema particular de los HMM (Problema 3).

Una de las versiones más eficientes del algoritmo Baum-Welch se llama algoritmo de checkpointing, el cual ejecuta en $O(NlogT)$ de memoria y $O(N^2TlogT)$ de tiempo.

3.3.3.2 Entrenamiento supervisado

Los modelos HMM pueden ser entrenados en modo supervisado en el caso de tener ejemplares disponibles de las distintas fases de la cadena de Markov. A pesar de no haber un algoritmo específico para ello, con el uso directo de métodos estadísticos se pueden calcular las distintas probabilidades necesarias para el ajuste del modelo a los ejemplares de entrenamiento. De este modo las matrices de transición y de observación pueden ser estimadas a partir de los distintos ejemplares de entrenamiento disponibles. Específicamente, el método se basa en el conteo de frecuencias, es decir, se cuenta el número de transiciones y emisiones de cada observación para cada uno de los estados de la cadena.

$$a_{ij} = \frac{a(i,j)}{\sum_{k=0}^{N} a(i,k)} \tag{3.34}$$

$$q_j(h) = \frac{e(j,v_h)}{\sum_{k=0}^{M} e(j,v_k)} \tag{3.35}$$

La ecuación (3.34) representa la probabilidad de transición tras el conteo de frecuencias de transición entre los estados i y j. En (3.35), se indica la probabilidad de que la observación v_h sea emitida en el estado j.

3.4 Conclusiones

Durante la tesis doctoral se han estudiado distintos mecanismos de Aprendizaje Automático para hacer frente al cálculo de la eficiencia de las respuestas y a la predicción de ataques multi-paso. La selección del método más apropiado no es fácil debido a que no hay reglas establecidas para decidir cuál es la mejor opción. En la toma de decisión influyeron las ca-

racterísticas de distintos algoritmos y el estado del arte referente a sistemas de seguridad que utilizan este tipo de técnicas. Como se especificó en la sección 2.6 sobre el estado del arte en este ámbito de la seguridad y teniendo en cuenta las propuestas que utilizan Aprendizaje Automático, se ha concluido que:

- Sobre el estado del arte de AIRS, hay una propuesta que utiliza aprendizaje automático basado en el sistema inmune. El sistema inmune biológico presenta ciertas características que lo hacen sumamente útil desde el punto de vista del procesamiento de información, como son: memoria, tolerancia a fallas, capacidad de aprendizaje, reconocimiento de patrones y procesamiento distribuido. A pesar del creciente número de trabajos realizados en esta área, aún existen muchas interrogantes por contestar, pues el funcionamiento del sistema inmune natural aún no es comprendido en su totalidad. En consecuencia, los modelos que existen en la actualidad suelen ser motivo de controversia y por ello se optó por el uso de una Red Neuronal Artificial como algoritmo bio-inspirado para el cálculo de la eficiencia de la respuesta.
- Estudiando los trabajos previos sobre predicción de pasos de ataque, se han analizado trabajos que se centraban en el uso del clasificador de Bayes, Redes Bayesianas o HMM. Comparando estos sistemas, se decidió utilizar los HMM ya que es un modelo más robusto que el clasificador de Naive Bayes y es un modelo simplificado de las Redes Bayesianas Dinámicas.

4

ARQUITECTURA

4.1 Introducción

Durante este capítulo se muestran las contribuciones propuestas para la tesis doctoral y la arquitectura desarrollada para mejorar los sistemas de protección contra intrusiones.

Como se analizó en el capítulo anterior sobre defensa perimetral, los Sistemas de Detección de Intrusiones han evolucionado rápidamente y a día de hoy hay gran variedad de herramientas maduras basadas en distintos paradigmas según sus principios de detección con un alto grado de fiabilidad (véase la sección 2.5.1). Los Sistemas de Prevención de Intrusiones han sido desarrollados como combinación de IDS con respuestas básicas, tales como resetear una conexión (véase sección 2.5.2). En cambio los Sistemas de Respuesta a Intrusiones (IRS) proporcionan los medios para lograr respuestas específicas de acuerdo con algunas reglas predefinidas como se menciona en mayor detalle en la sección 2.5.3 del estado del arte.

Como parte de la tesis doctoral, se propone la mejora del AIRS implementado en [ML13],

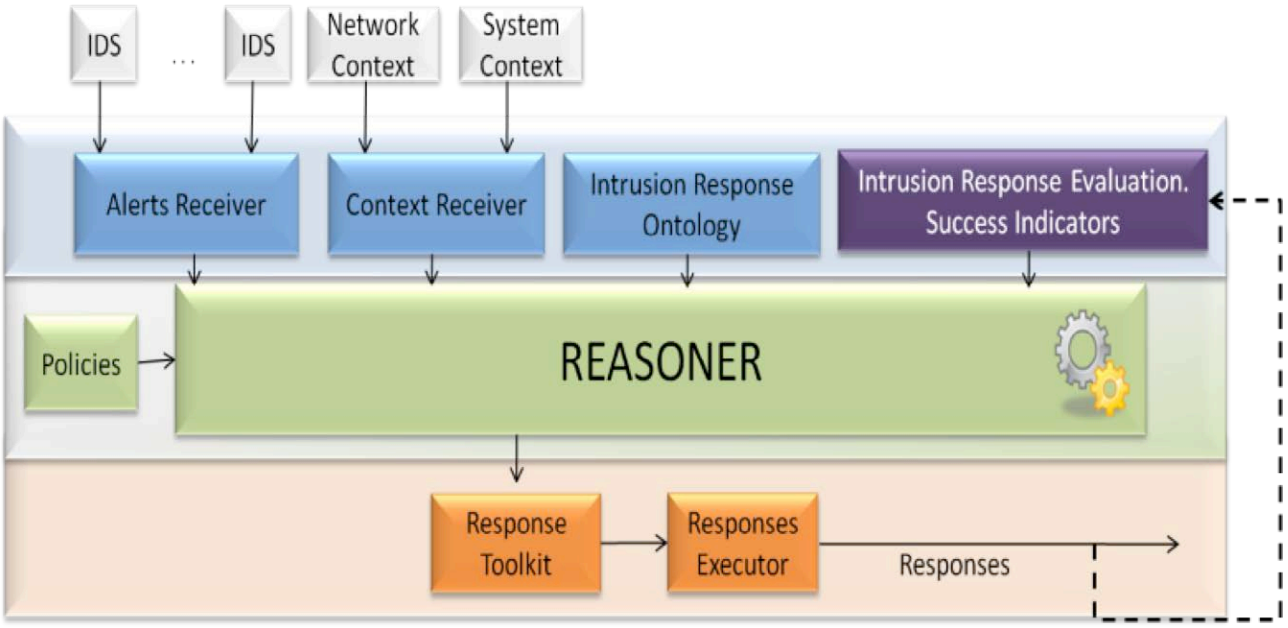


Figura 4.2.1: Arquitectura del AIRS [ML13]

utilizando técnicas de Aprendizaje Automático para consiguir un sistema predictivo ante ataques multi-paso, proactivo y con la capacidad de auto-aprendizaje. Por este motivo, antes de comenzar a explicar la arquitectura propuesta, se comentará en mayor detalle el AIRS donde se integran las contribuciones de la tesis doctoral.

4.2 Arquitectura del sistema de respuestas

La sección 2.6.2 menciona varios sistemas de respuestas a intrusiones automático desarrollados en investigaciones previas. El primero de ellos, desarrollado como parte del proyecto RECLAMO es el utilizado como sistema de respuestas para la integración de las contribuciones de la tesis doctoral. Este sistema se desarrolla basado en modelos formales de información definidos por ontologías [dVVM+09]. En concreto, el AIRS desarrollado se basa en una Ontología IDMEF para homogeneizar la información que proporcionan las alertas generadas por los eventos sospechosos de la red y almacenadas en clases de la Ontología [DVGVB09]. De este modo, es posible tomar las ventajas de la tecnología proporcionada por la Web Semántica, como es la inferencia de información.

La Figura 4.2.1 representa los módulos necesarios para la construcción del AIRS desarrollados para el proyecto RECLAMO [MLVGB10]. El objetivo de esta arquitectura es elegir la respuesta óptima a partir de un conjunto de respuestas disponibles. El AIRS recibe un conjunto de entradas, incluyendo el informe de la intrusión, la información del contexto, políticas de métricas de seguridad y la ontología de respuestas a intrusiones. Las políticas especifican distintas métricas que son elegidas dependiendo del contexto y el tipo de intrusión. A continuación se va explicar los módulos más importantes que componen la arqutectura:

- El razonador ejecuta procesos de inferencia para elegir la mejor respuesta basado en el resto de módulos: *Policies*, *Alerts Receiver*, *Context Receiver* e *Intrusion Response Ontology*. OWL (Web Ontology Language) [MVH+04] es el lenguaje de ontologías utilizado para definir toda la información necesaria para dicho proceso.

- La Ontología de respuestas a intrusiones define los conceptos y las relaciones necesarias para el sistema autónomo. La Ontología está basada en la estructura IDMEF, incluyendo clases y propiedades.

- El módulo de evaluación de la respuesta es el encargado de evaluar la respuesta lanzada por el AIRS. El método de evaluación de la efectividad de la respuesta está basada en la Entropía de Renyi del contexto [Shao1], pero tiene como inconveniente la cantidad de parámetros y umbrales que hay que determinar previamente. En concreto, se debe calcular el umbral alto y bajo para la efectividad de la respuesta considerando los siguientes factores:

  - La respuesta llevada a cabo: Cada respuesta tiene su propio impacto sobre el sistema comprometido. Por ejemplo una respuesta potente como cargar una honeynet debería tener un umbral más alto para ser considerado exitoso que una respuesta más simple como cerrar un puerto o bloquear una dirección IP sospechosa. Por ello el sistema debe calcular los umbrales para cada una de las respuestas que pueda ejecutar el AIRS.

– Requerimientos y políticas de seguridad del sistema objetivo. En un sistema crítico los umbrales serán más altos que para uno que no lo es. Se opta por dos parámetros *ModifyLow* y *ModifyHigh* para modificar los dos umbrales para cada respuesta. Son seleccionados por el administrador del sistema antes del proceso de evaluación.

4.3 Proceso de inferencia del AIRS

La eficiencia de la respuesta es un factor esencial para lograr un AIRS dinámicamente adaptativo. Este factor, entre otros, es utilizado en el proceso de inferencia realizado por el razonador del AIRS para la selección de la mejor respuesta (Figura 4.4.1). Las métricas utilizadas para la inferencia de la mejor respuesta han sido definidas y analizadas en [MVRB12]. El proceso de inferencia sigue las siguientes etapas:

1. Obtener información sobre las alertas que llegan a la red de la organización. Concretamente se obtiene información sobre el contexto del sistema, el contexto de red y el informe de los IDSs.

2. Inferir el conjunto de respuestas recomendado:

   (a) Si la intrusión es similar a alguna previa, se selecciona la respuesta ejecutada previamente en el caso de que la eficiencia calculada a partir de la Red Neuronal fuera satisfactoria.

   (b) En cualquier otro caso, las respuestas recomendadas son inferidas basado en las políticas, métricas, tipo de intrusión y los parámetros del contexto.

3. Selección de la respuesta óptima de acuerdo con la importancia del activo. Para activos críticos o significativos se considera la eficiencia de la respuesta medida con el sistema de evaluación sobre ejecuciones previas de cada una de las respuestas.

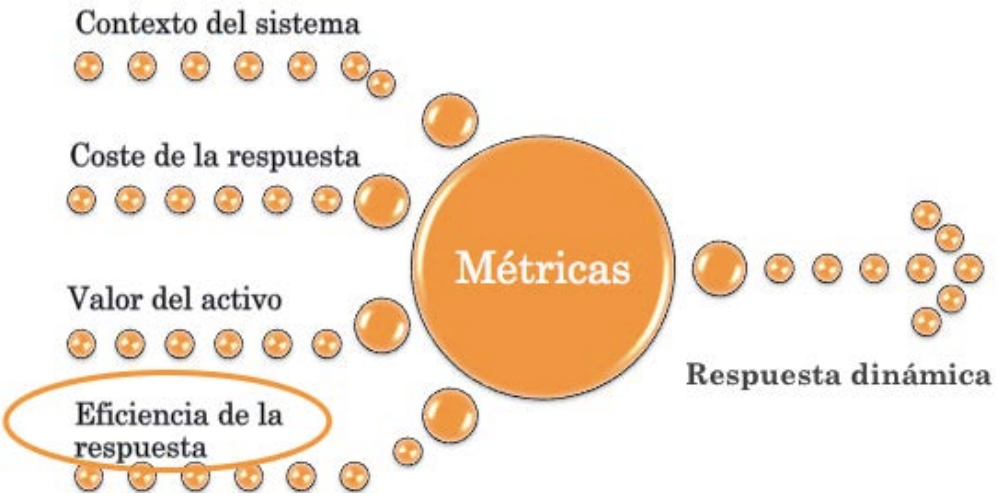


Figura 4.4.1: Esquema de obtención de la respuesta

4.4 Contribuciones

A continuación se detallan las contribuciones de la tesis doctoral para mejorar el sistema de respuestas desarrollado en [ML13]:

Contribución 1 Se propone conseguir la eficiencia de la respuesta usando una Red Neuronal Artificial tras una fase de entrenamiento previa, como mejora del método propuesto en [ML13]. Esta medida es vital para el AIRS ya que el valor de efectividad de la respuesta es utilizado en las métricas de selección de la respuesta para los activos críticos de la organización. En este sentido, proporciona auto-aprendizaje y dinamismo en las respuestas seleccionadas por el AIRS (Figura 4.4.1). Esta contribución es desarrollada en el capítulo 5.

Contribución 2 Proporcionar un mecanismo predictivo frente a ataques multi-paso para que el sistema de respuestas sea capaz de ejecutar respuestas proactivas para reducir la ventana de oportunidad del atacante y evitar en la medida de los posible daños en los activos de la organización. Esto es posible mediante la ejecución en paralelo de varios modelos HMM extendidos como se detalla en el capítulo 6.

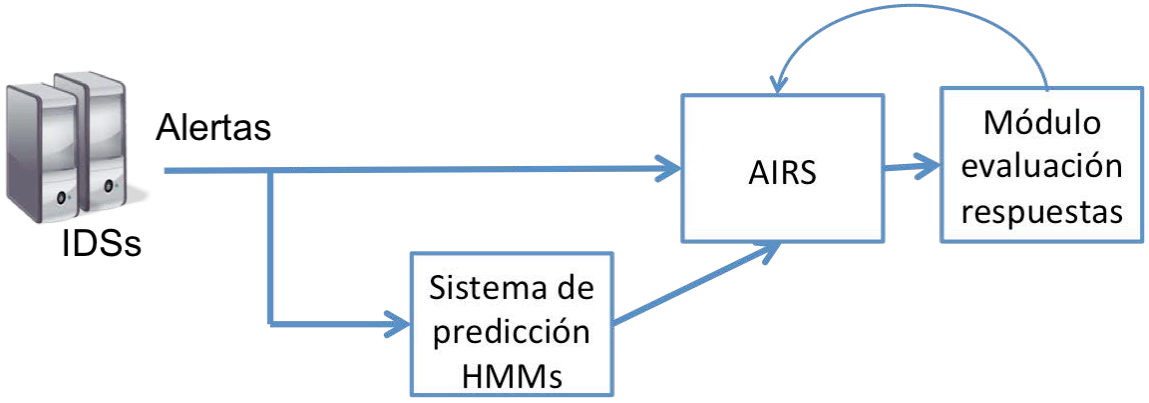


Figura 4.5.1: Arquitectura global de las contribuciones de la tesis doctoral

4.5 Arquitectura de mejora propuesta

La Figura 4.5.1 representa la arquitectura global simplificada del sistema de seguridad propuesto para la tesis doctoral. A continuación se explica cada una de las partes que componen dicha arquitectura:

- Los IDSs son los sistemas que monitorizan la red y los sistemas, en busca de indicios de ataque. Las alertas que producen estos sistemas son utilizadas como entrada del sistema propuesto. En concreto, se utiliza el formato IDMEF (Intrusion Detection Message Exchange Format) [DCF] para la representación de las alertas consiguiendo homogeneidad independientemente del tipo de IDS que reporte la alerta.

- El sistema de predicción de ataques multi-paso mediante HMMs es la principal contribución de esta tesis doctoral. Básicamente consiste en categorizar el ataque multi-paso en progreso, determinar la fase de ataque en la que se encuentra y calcular la probabilidad de que el ataque evolucione. Esto es posible mediante la ejecución en paralelo de varios modelos HMM extendidos como se detalla en el capítulo 6.

- El módulo de evaluación de la respuesta es el encargado de evaluar la eficiencia de la respuesta ejecutada para paliar o evitar los efectos de un ataque. En concreto se propone conseguir el cálculo de la eficiencia de la respuesta usando una Red Neuronal Artificial como segunda contribución de la tesis doctoral (véase el capítulo 5). El *feedback*

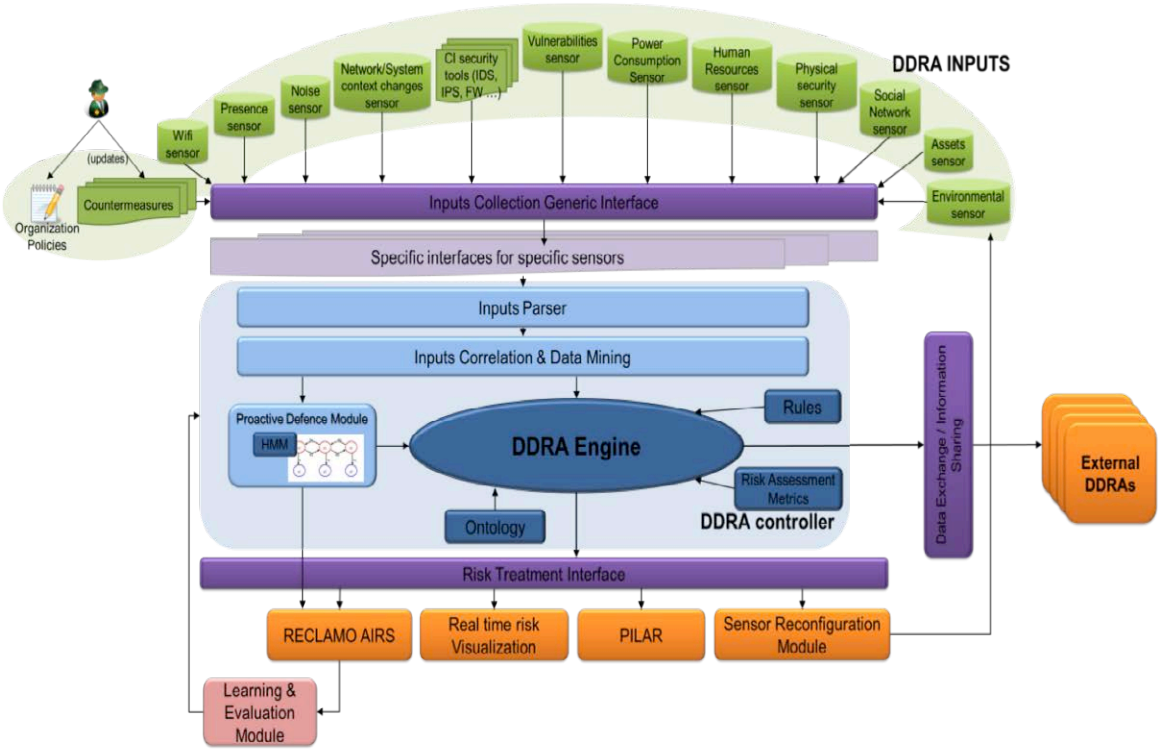


Figura 4.5.2: Arquitectura del proyecto DHARMA [DHA15]

obtenido del cálculo realizado sobre la eficiencia de la respuesta puede ser utilizado tanto por el sistema de respuestas como por el sistema predictivo.

- El AIRS se corresponde con el desarrollado en [ML13] en el proyecto RECLAMO, como se puede ver en la Figura 4.2.1.

Este esquema es validado dentro del entorno del proyecto DHARMA [DHA15], el cual se basa en la arquitectura de la Figura 4.5.2. En este caso, el sistema de predicción desarrollado en la tesis doctoral se utiliza tanto para las respuestas proactivas del AIRS como para proporcionar valores predictivos a un motor de Análisis de Riesgo Dinámico para que el valor del riesgo sea modificado dinámicamente. Por otro lado, el feedback proporcionado por la eficiencia de la respuesta puede ser utilizado por todo el conjunto del sistema de seguridad propuesto en DHARMA.

4.6 Conclusiones

Durante este capítulo se ha detallado la arquitectura global del sistema propuesto en la tesis doctoral. Este diseño proporciona la capacidad de predicción, proactividad y auto-aprendizaje a los sistemas de protección frente a ataques multi-paso, mejorando el margen de tiempo de reacción frente al ataque y por tanto evitando en gran medida el posible daño causado por el atacante a la organización.

En los siguientes capítulos se explica con mayor detalle el diseño de cada una de las partes que componen la arquitectura global, distribuidas en dos contribuciones de tesis.

5

APRENDIZAJE Y AUTOMATIZACIÓN PARA RESPUESTAS A INTRUSIONES

5.1 Introducción

Hoy en día, los IRSs están jugando un papel importante en la arquitectura de seguridad. Estos sistemas mitigan el impacto de los ataques con el objetivo de mantener la integridad, la confidencialidad y la disponibilidad de los recursos. En concreto, los Sistemas de Respuestas a Intrusiones Autónomos (AIRS) proporcionan la mejor defensa posible debido a que eliminan o mejoran el retardo ocasionado por el tiempo de demora de los administradores de seguridad en la toma de decisión sobre la respuesta a llevar a cabo contra el ataque. Para ello utilizan una serie de métricas definidas utilizando diferentes parámetros de medida necesarios para la selección de la respuesta.

La mayor parte de los AIRSs actuales tienen un enfoque fijo para las métricas de respuesta, es decir que las métricas no son modificadas de manera dinámica en función de distintos

factores externos. Dentro del ámbito del proyecto RECLAMO, se propone una arquitectura de seguridad capaz de seleccionar de manera dinámica la respuesta más apropiada, véase sección 4.2.

La contribución de la tesis doctoral a este AIRS desarrollado dentro del marco del proyecto RECLAMO es el análisis de la eficiencia de las respuestas ejecutadas cada vez que llega una alerta al sistema. Además, esta propuesta de tesis aporta la capacidad de auto-aprendizaje al AIRS a partir de respuestas lanzadas previamente. Con este objetivo se propone el uso de una Red Neuronal Artificial como mecanismo de *Machine Learning*. En concreto, la Red Neuronal propuesta debe clasificar el éxito o no de las respuestas lanzadas por el AIRS, de este modo, el sistema de respuestas incorpora un proceso de auto-aprendizaje a partir del ratio de éxito de la respuesta y así, poder responder mejor a futuros incidentes del mismo tipo.

5.2 Diseño de la Red Neuronal Artificial

Una de las contribuciones de la tesis se basa en la obtención de una medida del éxito de la respuesta con respecto a una intrusión concreta y para ello se propone diseñar una Red Neuronal Artificial debido a que son apropiadas para solventar problemas como el reconocimiento de patrones y la clasificación como se mencionó en la sección 3.2.2. A continuación se detalla el diseño de dicha Red Neuronal.

5.2.1 Topología

La topología de las Redes Neuronales, como ya se ha mencionado en la sección 3.2.3, está compuesta por lo menos de dos capas, una de entrada y otra de salida. Para el sistema propuesto la salida está compuesta por una sola neurona que representa la eficiencia de la respuesta para cada una de las respuestas lanzadas por el sistema cada vez que llega una intrusión.

Para determinar los parámetros de entrada de la Red Neuronal se han analizado distintas opciones a partir de la información de contexto obtenida a través del módulo *Context*

Receiver de la Figura 4.2.1:

- *Opción 1*: Los parámetros de entrada se podrían corresponder directamente con los parámetros obtenidos del contexto del sistema y la red tras la ejecución de la respuesta. En este caso las entradas de la Red Neuronal serían incluidas directamente de la salida obtenida del módulo *Context Receiver*.

- *Opción 2*: Los parámetros de entrada al algoritmo se podrían corresponder con el grado de anomalía del contexto del sistema y la red tras la respuesta. Este valor puede ser obtenido basado en la variación de la entropía incluida en [ML13].

La desventaja de la *Opción 1* radica en que los valores del contexto que podríamos determinar como fuera de ataque o atacado, son dependientes de la máquina o dispositivo en donde se está produciendo el ataque y del uso de dicha máquina dentro de la organización. Por tanto, para que el sistema sea totalmente independiente de la organización, es necesario que la entrada de la Red Neuronal se base en la *Opción 2*. Dentro de esta opción hay dos posibilidades que pueden ser aplicadas:

- *Opción 2a*: Grado de anomalía entre el *contexto normal* y el contexto tras aplicar la respuesta: Para esta solución es necesario obtener previamente el *contexto normal* en una fase de entrenamiento anterior al sistema en producción, quedando almacenado en una base de datos. El contexto tras aplicar la respuesta se obtiene en dicho momento a partir de los valores retornados por el módulo *Context Receiver*.

- *Opción 2b*: Grado de anomalía entre el contexto en ataque y el contexto tras aplicar la respuesta: En este caso, los valores del contexto serán obtenidos desde el módulo *Context Receiver* en el momento en que se detecta el ataque y tras la aplicación de la respuesta por parte del AIRS.

La Figura 5.2.1 muestra como varía el grado de anomalía de un valor de contexto, como por ejemplo podría ser la latencia del sistema. El valor *n* representa el máximo valor de grado de anomalía aceptado para considerar que el sistema está fuera de ataque o en estado

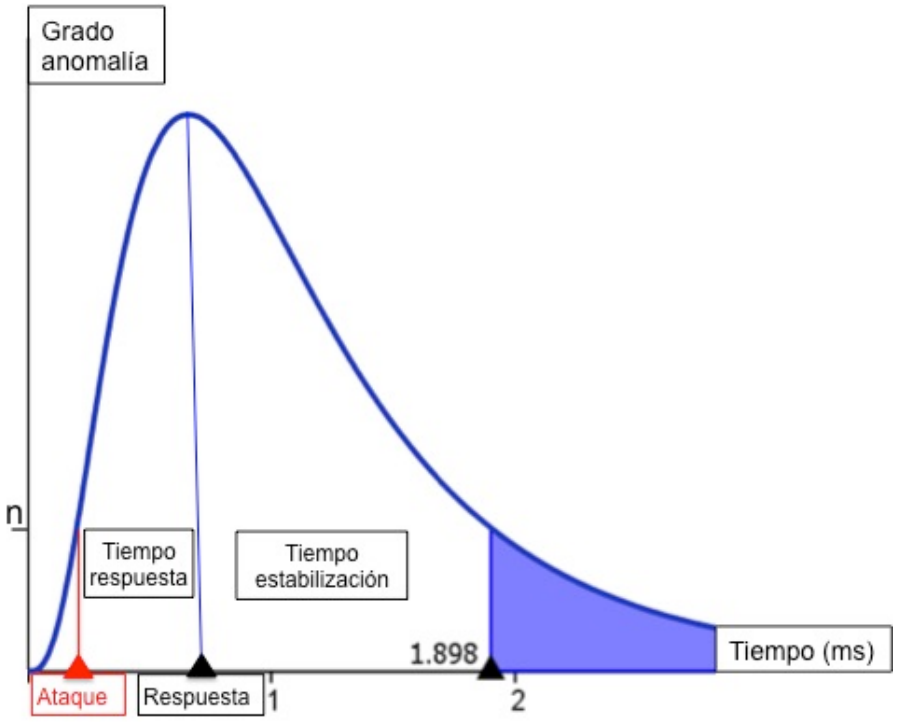


Figura 5.2.1: Ejemplo de variación en el grado de anomalía

normal. La línea roja de ataque representa el momento en que el AIRS considera que se está produciendo un ataque, por ejemplo un DoS, debido a las alertas reportadas por el IDS y el aumento del grado de anomalía del contexto. La primera línea azul de respuesta indica cuando se ejecuta la respuesta seleccionada por el AIRS. En el caso de que la Red Neuronal utilice como entradas la *Opción 2a*, la medida del *Contexto Normal* se obtiene de la base de datos y el valor de contexto después de la respuesta es medido tras la estabilización del sistema atacado, en el caso de la Figura 5.2.1 la medida del contexto se tomará a los 1.898ms. Para el caso de una Red Neuronal basada en la *Opción 2b* se tendria que medir el contexto tras la detección del ataque y tras la respuesta, que al igual que en el caso anterior se tomaría a los 1.898ms. Ambas opciones son válidas para que la Red Neuronal calcule la efectividad de la respuesta para cualquier activo de la organización, la diferencia radica en el número de solicitudes al módulo *Context Receiver* en producción y en si es necesario que previamente se almacene información del contexto de los activos de la organización. La decisión depende del entorno en el que se implante el sistema, siendo vital la capacidad de almacenamiento

y la velocidad a la que se extrae la información de contexto. Debido a que en el entorno de validación utilizado para la tesis doctoral, el módulo *Context Receiver* era lento, realizar dos solicitudes de parámetros de contexto para el cálculo de la efectividad de la respuesta afectaría al tiempo de respuesta del AIRS, debido que hasta que no tengamos el valor de contexto en ataque no es posible ejecutar la respuesta. Por lo que se optó por el almacenamiento de los parámetros de *contexto normal* de nuestro entorno de pruebas (emphOpción 2a), aunque esta opción repercuta en la necesidad de almacenamiento previo del contexto de todos los activos de la organización donde se quiera desplegar el sistema.

En concreto, se consideran siete parámetros del contexto del sistema (estado, latencia, uso de CPU, espacio de disco, número de procesos activos, número de usuarios y número de procesos *zombie*) y un parámetro que evalúa el contexto de la red. El grado de anomalía del contexto es calculado por la varianza de la entropía basada en la Teoría de Shannon y es aplicada a cada uno de los distintos ámbitos de contexto considerados. A continuación se muestra el grado de anomalía del contexto de *uso de CPU*, siendo igual para el resto de parámetros.

$$\Delta H_{CPU} = -\log \frac{Contexto_{Ataque}}{Contexto_{Normal}} \tag{5.1}$$

Hay que tener en cuenta que la varianza de la entropía calculada debe ser numérica y normalizada, ya que valores muy grandes pueden perjudicar la efectividad del algoritmo empleado. Debido a ello, como el rango de representación del grado de anomalía es un número entre 0 y 10, cada uno de estos valores será divididos entre 10 quedando un valor entre el rango $[0,1]$.

Durante esta etapa de diseño se ha considerado que es mejor no dejar fijado el número de capas ocultas y sus neuronas, debido a que es necesario una fase de experimentación con distintas topologías para determinar la optima para una Red Neuronal que calcule la efectividad de la respuesta basado en el contexto de la red y del sistema. Para ello la implementación llevada a cabo es totalmente parametrizable respecto al número de capas y de neuronas con el fin de encontrar la mejor topología durante la fase de experimentación y entrenamiento

de la Red Neuronal.

5.2.2 Selección del algoritmo

Para la evaluación de la respuesta se ha optado por el uso del algoritmo de Retropropagacion (backpropagation) o también llamado Perceptrón multicapa (LMP), véase sección 3.2.5. De este modo se lleva a cabo un entrenamiento supervisado aprovechando el conocimiento previo de intrusiones y respuestas apropiadas. En particular, el algoritmo se centra en determinar la satisfacción de la respuesta para cualquier sistema o intrusión. Esta decisión de diseño se ha basado en que el entrenamiento supervisado converge más rápido que utilizando algoritmos no supervisados debido a que el entrenamiento es guiado por un fichero de entrenamiento. Aunque como inconveniente es necesario tener un conjunto de instancias clasificadas durante la experimentación sobre el sistema en pruebas.

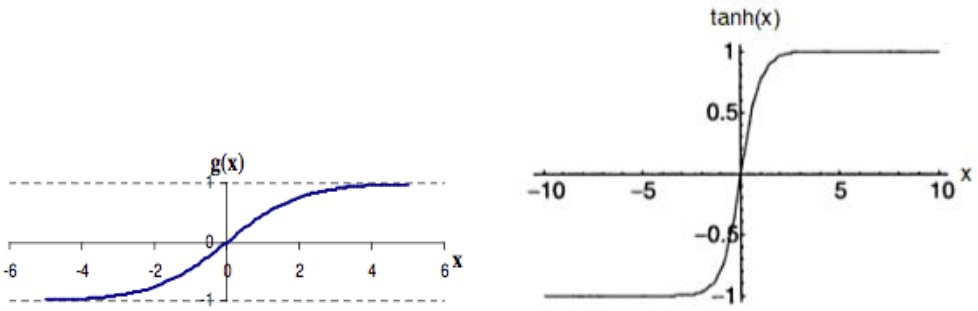
Antes de comenzar el proceso de entrenamiento es necesario la inicialización de los pesos asociados a cada neurona. En este caso se realiza una inicialización de pesos aleatoria. Se ha demostrado que puede funcionar bien el rango de $[-\beta, \beta]$ con $\beta = 0,7 \cdot \sqrt[n]{p}$; donde p es el número de unidades de entrada y n es el número de unidades de la capa oculta.

Por otro lado, se han implementado dos mejoras, tanto el Factor de aprendizaje adaptativo como el Aprendizaje con momentos, para evitar que el entrenamiento del algoritmo de Retropropagación converja en un mínimo local

5.2.2.1 Función de activación

Para la Red Neuronal propuesta se ha decidido implementar dos funciones de activación bipolares ya que estabilizan más rápido el error que con funciones binarias, disminuyendo el número de épocas en el proceso de entrenamiento. En concreto se ha implementado la función sigmoide (Figura 5.2.2a) y la función tangente hiperbólica (Figura 5.2.2b), ambas bipolares, con lo que la salida queda acotada como un valor numérico real entre [-1,1].

De todas las funciones de activación se ha decidido implementar estas dos debido a que, la función sigmoide minimiza la capacidad de cálculo necesario en la fase de entrenamiento



(a) Función sigmoide bipolar **(b)** Función tangencial hiperbólica

Figura 5.2.2: Funciones de activación [KO11]

y la función tangente hiperbólica es la que proporciona mayor precisión en los resultados [KO11].

5.2.2.2 Condición de parada

Como condición de parada del algoritmo de Retropropgación se utiliza el error cuadrático medio (ECM) o en inglés *Mean Square Error* (5.2), donde y_{obj} es la salida deseada e y_{en} se corresponde con la salida generada por el algoritmo para una entrada concreta.

Este método se basa en encontrar los pesos que minimicen el ECM con el objetivo de conseguir la mayor precisión en la clasificación. Por otro lado, para evitar que el algoritmo no encuentre en un tiempo razonable el error mínimo considerado, también se añade como condición de parada un número máximo de épocas específico.

$$ECM = \sum (y_{obj} - y_{en})^2 \tag{5.2}$$

5.2.3 Validación cruzada

La validación es la ultima fase del desarrollo de la propuesta de tesis doctoral y es la etapa que se encarga de determinar si es necesario entrenamiento adicional con una mayor cantidad de patrones o por el contrario es mejor optar por otro tipo de topología para la Red Neuronal.

Para la contribución de tesis se ha optado por la realización de la *validación cruzada* mencionada en la sección 3.2.6 para evitar sobreajuste de la Red Neuronal. Tanto el conjunto

```
###########Fichero de configuracion de la Red Neuronal#############
#Numero de entradas
entradas=3
#Numero de salidas
salidas=1
#Funcion de activacion
activacion=sigmoid
#Numero de capas ocultas
capas=1
#Numero de neuronas en cada capa oculta (separadas por espacios)
ocultas=1
#valor de bias
bias=1.0
#Pesos de cada una de las conexiones ordenadas por neuronas de cada capa
# Ejemplo: [peso_x1 ...peso_xn peso_bias][peso_x1 ...peso_xn peso_bias]
[0.5379116352292687 0.6937946787749012 0.045935214904464367 0.9967865649026152 ]
[-0.12500687660498866 0.36593397916382786 ]
```

Figura 5.2.3: Configuración de una Red Neuronal con función de activación sigmoide logística bipolar

de entrenamiento como el de test estarán compuestos por una mezcla entre casos de éxito (salida = 1) y fracaso (salida = -1) Para la realización de dichos conjuntos necesitamos ejecutar para todas las posibles respuestas una batería de intrusiones de las cuales conozcamos la eficacia o no de cada una de las respuestas e ir obteniendo las variaciones de contexto. Una vez que tengamos todas las muestras es necesario separarlas en conjunto de entrenamiento y de test para comprobar la robustez de los pesos. De este modo es posible conseguir la generalización de los resultados.

El formato de los ficheros utilizados para la validación está compuesto por una serie de líneas, una por cada alerta generada por el IDS. Cada una de estas líneas contiene los valores de grado de anomalía del contexto (VE1 VE2 VE3 VE4 VE5 VE6 VE7 VE8) obtenidos desde el módulo *Context Receiver* para cada uno de los parámetros de contexto del sistema y la red; y el valor de salida (VS1) donde se indica con el valor -1 las respuestas erróneas y con el valor 1 las correctas.

5.2.4 Fichero de configuración de la Red Neuronal

La Figura 5.2.3 y la Figura 5.2.4 son dos ejemplos de fichero de configuración que representan una Red Neuronal, tanto utilizando como función de activación la sigmoide logística bipolar como utilizando la función de activación tangente sigmoidal hiperbólica respectivamente.

```
##########Fichero de configuracion de la Red Neuronal############

#Numero de entradas
entradas=2

#Numero de salidas
salidas=1

#Funcion de activacion
activacion=tangencial

#Numero de capas ocultas
capas=1

#Numero de neuronas en cada capa oculta (separadas por espacios)
ocultas=2

#valor de bias
bias=1

#Pesos de cada una de las conexiones ordenadas por neuronas de cada capa
# Ejemplo: [peso_x1 ...peso_xn peso_bias][peso_x1 ...peso_xn peso_bias]
[0.7 -0.2 0.4][-0.4 0.3 0.6]
[0.5 0.1 -0.3]
```

Figura 5.2.4: Configuración de una Red Neuronal con función de activación tangente sigmoidal hiperbólica

A continuación se detalla cada una de las partes del fichero de configuración:

- Número de entradas: Indica el número de neuronas de la capa de entrada de la Red Neuronal. En el caso referente a la tesis doctoral, esta capa debe tener 8 neuronas, una por cada valor de grado de anomalía del contexto.
- Número de salidas: Se corresponde con el número de neuronas de la capa de salida de la Red Neuronal. Para el cálculo de la efectividad de la respuesta, solo hay una neurona de salida correspondiente al valor de efectividad de la respuesta ejecutada por el AIRS.
- Función de activación: Hay dos valores posibles:
  - sigmoide: Referencia al uso de la función de activación sigmoide logística bipolar.
  - tangencial: Hace referencia al uso de la función de activación tangente sigmoidal hiperbólica.
- Número de capas ocultas: Indica el número de capas ocultas utilizadas en la topología de la Red Neuronal.

- Número de neuronas en cada capa oculta: Se debe indicar el número de neuronas de cada una de las capas ocultas establecidas.
- Bias: Valor numérico de la entrada *Bias* presente en todas las capas de entrada y ocultas de la Red Neuronal.
- Pesos de cada una de las conexiones: A partir de este momento, se van indicando cada uno de los pesos de las conexiones entre las neuronas que componen la topología de la Red Neuronal establecida siguiendo el siguiente formato por cada neurona $[peso_{x1}...peso_{xn}, peso_{bias}]$ e incluyendo una nueva línea por número de capas ocultas y de salida.

5.3 Integración en el sistema de respuestas

La Ontología de respuestas a intrusiones incluida en la sección 4.2 define los conceptos y las relaciones necesarias para el sistema autónomo. La Ontología está basada en la estructura IDMEF, incluyendo clases y propiedades. Hay dos clases principales relacionadas con el sistema de evaluación: *Response* y *Result*. Cada una de estas clases tiene una serie de propiedades relacionadas con este objetivo. En concreto para la clase *Response* hay dos propiedades asociadas, *executionTimes* y *sucessFactor* . La propiedad *responseEfficiency* está incluida en la clase *Result*. Estas propiedades serán las que jueguen un papel importante en la realimentación del aprendizaje desarrollado como propuesta de tesis.

Una vez se ha llevado a cabo el proceso de entrenamiento de la Red Neuronal, se conocerán los pesos y la topología que mejor encajan con la evalución de la eficiencia de la respuesta ejecutada en el momento del ataque. Una respuesta satisfactoria tendrá un valor cercano o igual a 1 y una respuesta no satisfactoria tendrá un valor cercano o igual a -1. Siendo la salida de la Red Neuronal un valor real dando mayor precisión al resultado. Este valor se corresponderá con el parámetro *sucessLevel* para el cálculo de la eficiencia de la respuesta con las

siguientes ecuaciones:

$$sucessFactor = \sum_{j=0}^{j-1} sucessLevel_i \tag{5.3}$$

$$responseEfficiency = \frac{sucessFactor}{executionTimes} \tag{5.4}$$

Como se ha mencionado antes, los parámetros *executionTimes*, *sucessFactor* y *responseEfficiency* son valores incluidos en la Ontología del AIRS. En concreto *executionTimes* y j se corresponden con el número de veces que la respuesta ha sido ejecutada.

5.4 Proceso de evaluación de la Red Neuronal Artificial

La Red Neuronal necesita una fase previa de entrenamiento para seleccionar la topología más apropiada y obtener los mejores pesos de las conexiones entre las neuronas. Para facilitar esta tarea la implementación se ha llevado a cabo de manera totalmente parametrizable. De este modo se permite la estudiar la propuesta desde varios puntos de vista, incluido el de la eficiencia de las distintas topologías y así determinar si es viable la adaptabilidad del AIRS en tiempo real.

Para llevar a cabo la validacións se optó por el uso de un entorno virtual controlado dentro del ámbito del proyecto RECLAMO. Como se puede ver en la Figura 5.4.1 el escenario simula la red de una organización con diferentes servidores, firewalls y hosts. Este escenario ha sido desarrollado utilizando la herramienta VNX [GFDVC10].

Una vez realizada la integración del módulo de evaluación en el AIRS desarrollado en el proyecto RECLAMO se debe llevar a cabo varias etapas:

1. Recopilación de los valores de *contexto normal* en un entorno controlado, es decir, sin ataques que perturben los resultados. El prototipo de validación utiliza las herramientas *Nagios* y *Sancp* para la monitorización y recopilación de los ejemplares necesarios de contexto de sistema y de red. Para recoger datos realistas de los sistemas es necesario la generación de scripts que generen tráfico ficticio en cada uno de los hosts y

almacenar los datos en la base de datos MySQL preparada para guardar los datos del contexto de todos los activos en un estado *normal*.

2. Ejecución de una batería de ataques de distintos tipos para recoger los patrones necesarios para el entrenamiento de la Red Neuronal. Mediante la implementación de un script de automatización de los ataques generados sobre la red virtual con el objetivo de obtener los ejemplares necesarios para el entrenamiento y validación de distintas topologías de Redes Neuronales. Para la realización de ataques a las máquinas virtuales se utiliza la distribución *Kali Linux* y distintos módulos y scripts de ataque. Por ejemplo para la realización de un DoS se utiliza el script *slowloris*[KZD+15]. Además, cuando se ejecuta cada uno de los ataques, el script fuerza la ejecuci'on de las distintas respuestas posibles, ya sean o no válidas para dicho ataque, para obtener todas los posibles valores de contexto, de este modo la Red Neuronal aprenderá a diferenciar entre las respuestas buenas y malas frente a cada uno de los ataques.

3. Análisis de varias topologías para la Red Neuronal una vez tenemos disponibles todos los patrones necesarios para llevar a cabo la validación cruzada para seleccionar la mejor de todas para el caso de evaluación de la eficiencia de las respuestas.

4. Validación final del AIRS dinámicamente adaptativo con la Red Neuronal seleccionada representada por varias funciones matemáticas basadas en el algoritmo de Retropropagación y los pesos obtenidos tras la fase de entrenamiento y teniendo en cuenta las fórmulas explicadas en la sección 5.3.

Como primera topología a evaluar se propone una Red Neuronal de una sola capa oculta con ocho neuronas como se puede ver en la Figura 5.4.2. Las neuronas de la capa de entrada se corresponden con los valores de contexto de sistema y de red como se indicó en la sección 5.2.1. La salida es la efectividad de la respuesta obtenida a partir de la función de activación seleccionada en el fichero de configuración (véase la Sección 5.2) El número de ejemplares necesarios para el entrenamiento de la Red Neuronal propuesta es de unos 200

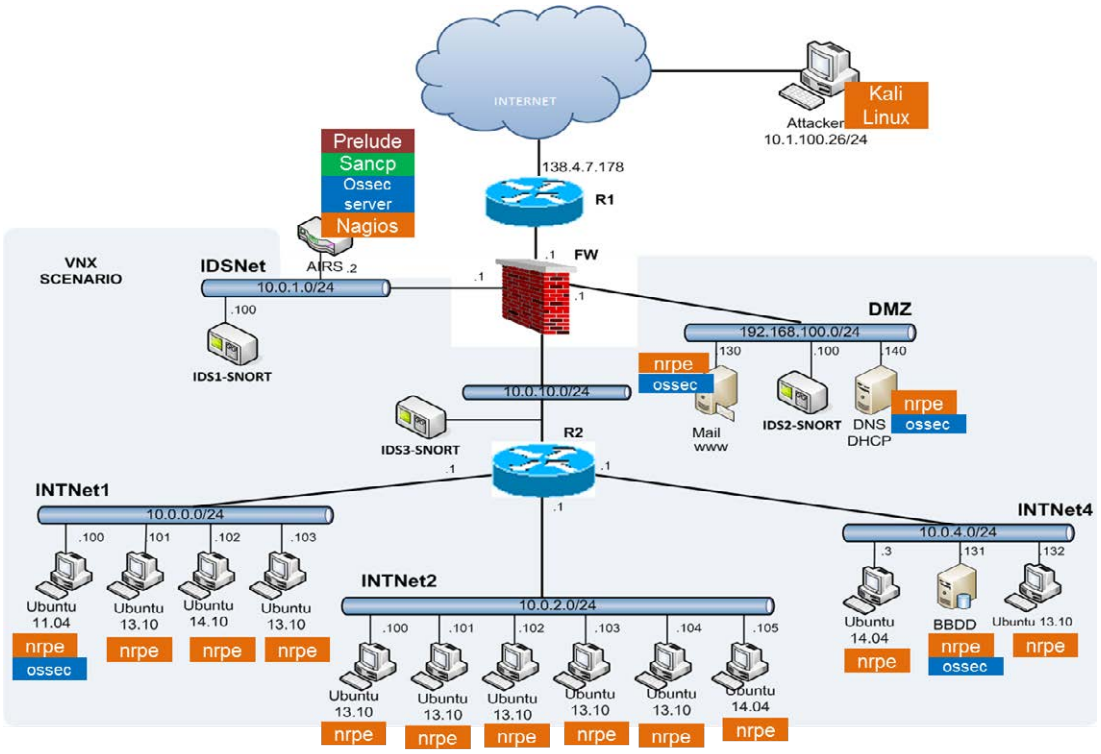


Figura 5.4.1: Escenario virtual de pruebas para el AIRS

ejemplares, incluyendo distintos tipos de parejas de ataque y respuesta obtenidos en la etapa 2 del proceso de evaluación explicado anteriormente.

Una vez realizado el proceso de entrenamiento de la red propuesta en la Figura 5.4.2 con el algoritmo de Retropropagación (véase la sección 3.2.5.1) se obtienen los valores de los pesos de cada neurona de entrada, w_{ij}, y de los pesos de cada una de las neuronas de la capa oculta, v_{jk}. Tras validar la Red Neuronal obtenida mediante el método de Validación Cruzada, la red estará preparada para llevar a cabo el cálculo de la efectividad de las respuestas ejecutadas por el AIRS a través de una serie de ecuaciones, las cuales podemos dividir en dos etapas:

Cálculo de propagación de activaciones desde la capa de entrada a la capa oculta: La ecuación (5.5) calcula el valor de propagación los 8 valores de entrada teniendo en cuenta los pesos de cada una de las neuronas de entrada, donde w_{oj} es el peso de la neurona de bias de entrada conectada a la neurona de la capa oculta z_j, x_i representa la neurona i de la capa de entrada, la cual se corresponde con uno de los valores de variación del contexto calculado a partir de

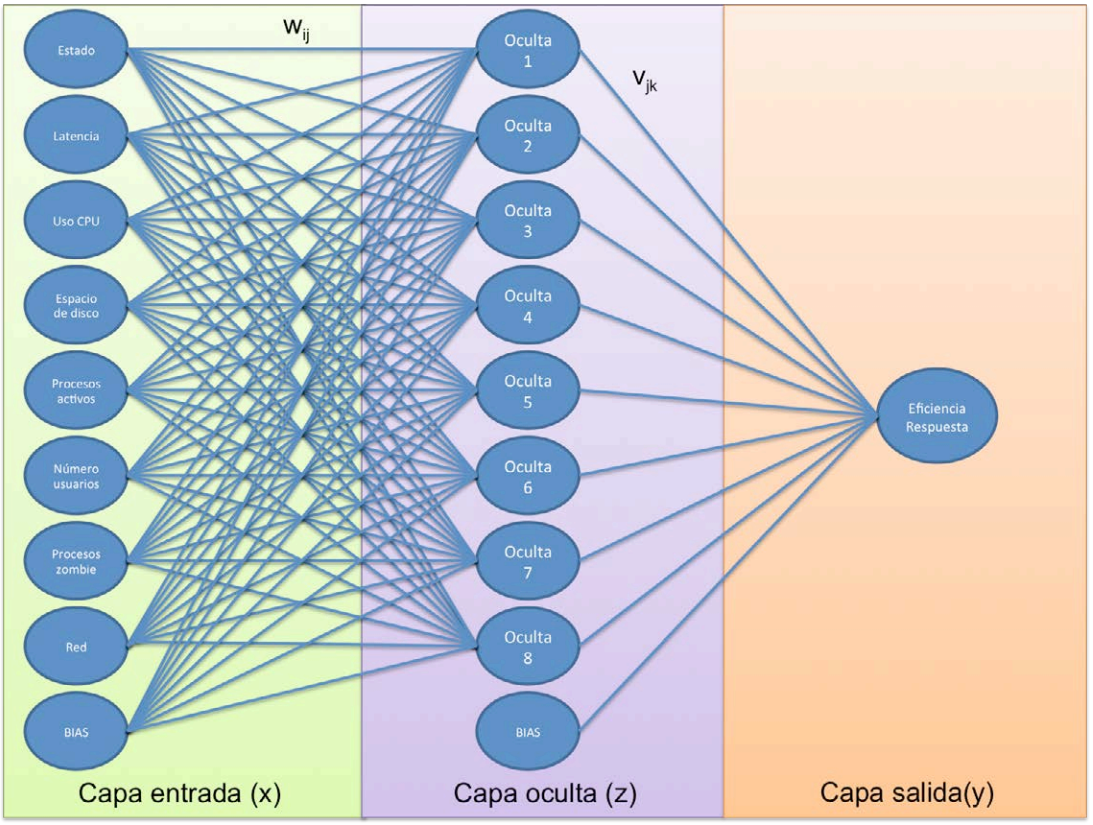


Figura 5.4.2: Red Neuronal propuesta

las medidas obtenidas en el momento en que se ejecuta la respuesta, y w_{ij} son los pesos de cada una de las neuronas de la capa de entrada conectadas a la neurona z_j de la capa oculta.

En la (5.6) se obtiene el valor final de la neurona aplicando la función de activación seleccionada en el fichero de configuración sobre el valor de propagación de la entrada z'_j.

Todos estos cálculos son llevados a cabo para cada una de las 8 neuronas z_j de la capa oculta.

$$z'_j = w_{oj} + \sum_{i=1}^{8} x_i \cdot w_{ij} \tag{5.5}$$

$$z_j = f(z'_j) \tag{5.6}$$

Cálculo de propagación de activaciones desde la capa oculta a la capa de salida: La ecuación (5.7) calcula el valor de propagación de cada uno de los 8 valores de entrada a la salida, teniendo en cuenta los valores de las neuronas de la capa oculta y sus pesos. v_{ok} es el peso de la neurona de bias de la capa oculta conectada a la neurona de salida y_k, z_j representa la neurona j de la capa oculta y v_{jk} son los pesos de cada una de las neuronas de la capa oculta conectadas a la neurona y_k de la capa de salida.

En la (5.8) se obtiene el valor final de la salida de la neurona y_k aplicando la función de activación seleccionada en el fichero de configuración sobre el valor de propagación y'_k.

Como se ha podido ver, para calcular la propagación desde las capas ocultas a la salida se aplica el mismo método que en el anterior caso, pero al tener solo una neurona de salida este proceso se realizará una sola vez para k=1.

$$y'_k = v_{ok} + \sum_{j=1}^{8} z_j \cdot v_{jk} \tag{5.7}$$

$$y_k = f(y'_k) \tag{5.8}$$

5.5 Conclusiones

Durante este capítulo se ha detallado el diseño y la implementación de un sistema capaz de calcular la eficiencia de la respuesta ejecutada por un AIRS a partir de una Red Neuronal.

En concreto se utiliza el algoritmo supervisado de retropopagación con funciones sigmoidales bipolares para la obtención del valor de la eficiencia de la respuesta en el momento de su ejecución por parte del AIRS. La fase de entrenamiento de la red y la elección de la mejor topología es la parte esencial para el buen comportamiento del sistema.

Finalmente, se detallan las fases necesarias para la validación de la Red Neuronal Artificial propuesta. Básicamente se compone de una primera fase de entrenamiento y de selección de la topología apropiada para después poder verificar el cálculo de la eficiencia de la respuesta en el AIRS.

Con este método es posible conseguir que el AIRS sea dinámicamente adaptativo y por

tanto aprenda a partir de respuestas previas a distintos ataques.

6

SISTEMA DE PREDICCIÓN DE ATAQUES MULTI-PASO

6.1 Introducción

La segunda contribución de este trabajo de tesis es la propuesta de un método novedoso para la predicción de ataques multi-paso basado en Modelos de Markov Ocultos (HMM).

Un escenario de ataque o ataque multi-paso es un conjunto de acciones realizadas por un atacante para conseguir un objetivo específico (véase sección 2.3). La predicción de ataques multi-paso es esencial para las organizaciones debido al incremento de ciberataques. La Denegación de Servicio Distribuida (DDoS) es uno de estos tipos de ataque más preocupantes a día de hoy con numerosos ejemplos en diversas organizaciones [Cer14] [Dyn16]. Pero no es el único, Microsoft antimalware telemetry detectó en Mayo de 2017 un nuevo ransomware denominado WannaCry mediante el uso de un modelo predictivo [Mic17]. Estos ataques pueden considerarse como ataques en varias etapas como se menciona en la sección 2.3.

En concreto, cada HMM representa un tipo distinto de ataque multi-paso, basado en una serie de fases similares que componen los estados ocultos de la cadena, representando las distintas acciones realizadas por el atacante. Como resultado la cadena de Markov se adapta al ataque multi-paso en curso, haciendo posible la predicción del objetivo final y del siguiente paso del atacante.

A continuación se detalla la arquitectura y la definición de modelos realizada como trabajo de tesis doctoral que consiste en procesar varios hilos o procesos de varios modelos que representan distintos ataques multi-paso basados en una definición extendida de HMM [HVV17].

6.2 Arquitectura

La Figura 6.2.1 representa la arquitectura implementada durante el desarrollo de la tesis doctoral para la predicción de ataques multi-paso. A continuación se explica cada módulo en mayor detalle:

- Los ficheros de ataque multi-paso (Multistep attack files) son ficheros de trazas *pcap* (packet capture) con las distintas alertas reportadas por el IDS de las intrusiones conocidas de cada fase de un ataque concreto. Estos ficheros son necesarios para el entrenamiento de los distintos HMM de cada tipo de ataque.

- El módulo de entrenamiento off-line (Off-line training module) representa la etapa previa de entrenamiento de los parámetros de cada HMM. Primero es necesario la obtención de las observaciones de cada una de las alertas almacenadas en los Multistep attack files. En la sección 6.4 se explica el proceso en mayor detalle.

- Los ficheros de configuración de los HMMs (HMM configuration files) almacenan los parámetros de cada uno de los HMMs tras el entrenamiento con los distintos escenarios de cada tipo de ataque multi-paso. En concreto, estos ficheros incluyen las matrices de probabilidad, el número de estados de la cadena de Markov etc.. Estos ficheros son necesarios para la predicción de los distintos escenarios de ataque.

- El módulo de predicción (Prediction module) aplica algoritmos específicos de los Modelos de Markov Ocultos a la secuencia de alertas que recibe desde los IDSs usando cada uno de los distintos HMMs configurados. Como resultado se obtiene la secuencia del ataque que está sucediendo, las fases de ataque que están aun por suceder y la probabilidad de que el atacante llegue a su objetivo. En las siguientes secciones se explica en mayor detalle.
- La probabilidad de ataque calculada puede ser utilizada en distintos ámbitos, como en la ejecución de respuestas proactivas de los AIRS o en Sistemas de Gestión de Riesgo Dinámico. El módulo de procesamiento proactivo (Proactive Processing Module) puede incluir cualquiera de estos sistemas.
- La base de datos de seguridad (Securitytroyano database) almacena todos los nombres de CVE y su correspondiente etiqueta seleccionada a partir de la descripción dicha vulnerabilidad. Esto será detallado en la sección 6.6.1.1.
- El módulo de coordinación de alertas (Alerts Coordination Module) realiza tanto la agregación como el filtrado de alertas desde diferentes IDSs. Además determina si las alertas pertenecen al ataque en curso o es el comienzo de un nuevo ataque. Con este módulo se podría reducir el número de falsos positivos basado en el nivel de confianza en cada IDS. Este módulo será diseñado como trabajo futuro.

6.3 Definición del Modelo de Markov Oculto

Cada HMM definido en la tesis doctoral se compone por dos procesos estocásticos, como se detalla en la sección 3.3. En el modelo propuesto, el proceso estocástico oculto está representado por una cadena compuesta por las diferentes fases o pasos de un tipo concreto de ataque multi-paso, mientras que las observaciones son obtenidas a partir de las alertas producidas por los atacantes.

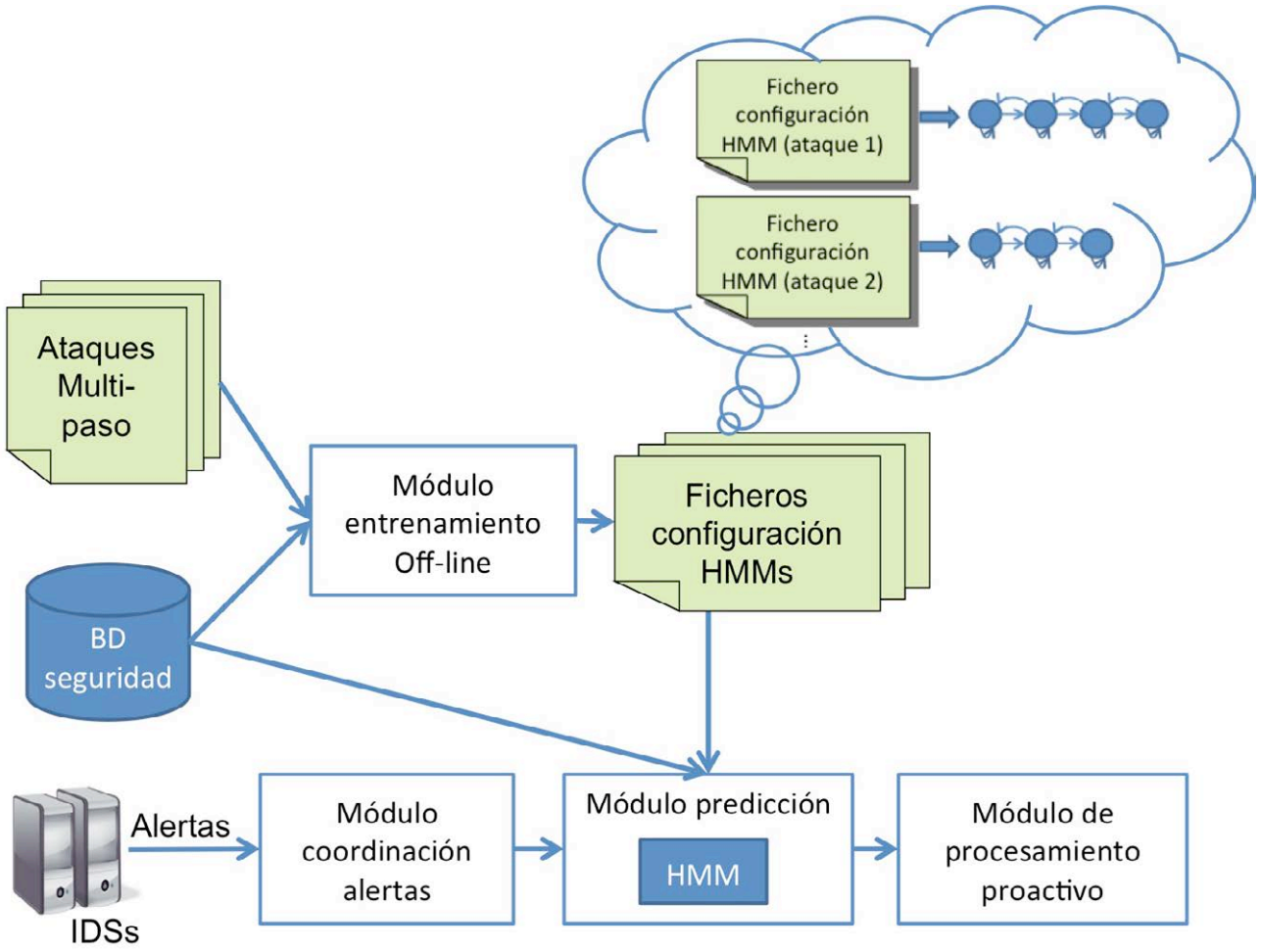


Figura 6.2.1: Arquitectura del sistema de predicción de ataques multi-paso

En concreto los modelos HMM usados en la presente tesis doctoral se describen con los siguientes procesos estocásticos:

1. Un proceso oculto que se corresponde con las distintas fases comunes de un tipo de ataque multi-paso y representado por los estados de la cadena de Markov, $x(t)$.

2. Un proceso de observaciones obtenidas a partir de las alertas que van llegando desde los IDSs a cada instante de tiempo t, $y(t)$.

En este caso, t representa tiempo discreto correspondiendo con las alertas que van llegando desde los IDSs.

Como aportación de la tesis doctoral, el sistema de predicción utiliza un Modelo de Markov Oculto extendido para cada escenario de ataque multi-paso en estudio, el cual se define como $\lambda_k = (\Sigma, S, A, Q, \pi, H)$. A continuación se explica cada uno de los parámetros que componen el modelo:

1. λ_k representa el modelo para cada ataque multi-paso, donde k es un número que identifica el tipo de ataque. De esta manera, se obtiene un modelo para cada tipo de ataque ajustado a las características de cada una de sus etapas, al contrario que otras propuestas analizadas durante el estado del arte (sección 2.6.3).

2. Σ representa todas las posibles observaciones basadas en las alertas de los IDSs y del reporte de CVE almacenado en la base de datos referenciada en la sección de 6.2. El conjunto de observaciones viene representado con la expresión (3.12).

3. S es el conjunto de estados que componen el HMM (3.13). En nuestro modelo estos estados se corresponden con los pasos que componen las etapas de los ataques de una misma tipología.

4. A es la matriz de probabilidad de transición de estados calculada en función de las alertas almacenadas para cada escenario de ataque. Debido a que la cadena de Markov está totalmente conectada entre estados de dicha cadena, el tamaño de la matriz es NxN. De esta forma es posible conocer las probabilidades de un estado con el resto de los que componen la cadena. Esta matriz se representa con la ecuación (3.14) donde i y j representan un estado del conjunto S.

5. Q es la matriz de probabilidad de observación,compuesta por un vector de probabilidad q_i para cada estado. Este vector indica la probabilidad de que distintos tipos de alertas puedan ser observadas por el modelo estando en un estado concreto. En la ecuación (3.15), i representa un estado del conjunto S y j se corresponde con una observación del conjunto Σ.

6. π es el vector de distribución inicial (3.16), el cual indica la probabilidad de que cada uno de los estados se corresponda con el estado inicial del ataque representado por el

modelo, es decir, la probabilidad de que el ataque se inicie en cada uno de los distintos estados de la cadena de Markov.

7. H es un vector que se incluye a la definición formal del HMM descrita en 3.3. Este vector representa el número medio de observaciones que pueden ser observadas en cada uno de los estados de la cadena de Markov. En concreto, dicho vector cumple la expresión $H = \{h_1, h_2, \ldots, h_N\}$, donde h_i representa el número medio de observaciones del estado i. Este vector es necesario para el cálculo de la probabilidad final de ataque (véase la sección 6.5.2).

Una vez definido el modelo, el sistema propuesto en la tesis doctoral lleva a cabo el entrenamiento para cada tipo de ataque multi-paso de manera *off-line*. Como resultado, el sistema propuesto almacena un fichero de configuración para cada tipo de ataque multi-paso (véase la sección 6.2). El fichero de configuración contiene los valores de todos los parámetros que componen el modelo λ_k del HMM para una tipología de ataque específica. De esta manera la probabilidad calculada se ajusta mejor al tipo de ataque que se está modelando, al contrario que en trabajos anteriores (véase la sección 2.6.3). El formato del fichero de configuración del HMM está basado en el formato propuesto en librería de Java *jahmm* [jah].

La Figura 6.3.1 muestra un ejemplo de un fichero de configuración con nuestro diseño de HMM:

- La primera línea indica el modelo matemático utilizado, el identificador y el nombre del ataque.
- A continuación se almacena el número de estados del conjunto S (*NbStates*) y el número de observaciones del conjunto Σ (*NObservations*)
- *Nevents* representa el vector que almacena el número medio de observaciones para cada estado.
- A partir de este momento, el fichero contendrá una serie de valores para cada estado de la cadena de Markov:

```
Hmm 2:Ataque de prueba

NbStates 2
NObservations 18
NEvents 27 47

State:Nombre estado 1
Pi 0.9999999999999994
A 0.9851440148946234 0.007427992552688263
DiscreteOPDF 0.02228397765806478 0.0 0.13946447060254585 0.0 0.0 0.0 0.0 0.0 0.006316385838303897 5 0.0 0.0
0.0 0.0 0.831935165901084 0.0 0.0 0.0 0.0
State:Nombre estado 2
Pi 0.0
A 0.5 0.5
DiscreteOPDF 0.0 0.0 0.013255590361844812 0.0 0.0 0.0 0.02165108474952669 0.4979749492391142
0.012445595909454123 0.0 0.010825542374763346 0.0 0.0 0.0 0.3788939831167173 0.0 0.06495325424858008 0.0
```

Figura 6.3.1: Fichero de configuración de un HMM de ejemplo

- *State: Nombre del estado*
- *Pi* que almacena el valor de probabilidad de que ese estado sea el inicial del ataque. Uniendo estos valores para todos los estados se obtiene el vector de distrinución inicial, π.
- *A* incluye los valores de probabilidad de transición desde ese estado al resto de los estados de la cadena. Uniendo todos estos valores de cada uno de los estados del modelo se obtiene la matriz de probabilidad de transición.
- *DiscreteOPDF* es el tipo de dato de la librería *jahmm* utilizado para las observaciones. Esta librería tiene varios tipos de datos para la representación de observaciones (Enteros,Discretos, Reales, ...). Concretamente, para el desarrollo de la propuesta de tesis doctoral se ha empleado observaciones discretas (véase 6.4.1). A continuación, aparecen una serie de valores que representan el vector de probabilidad de observación de dicho estado, q_i.

Las observaciones obtenidas a cada instante de los IDSs para un tipo de ataque multi-paso específico es almacenado en la secuencia de observaciones $O = \{o_1, o_2, \ldots, o_T\}$, donde cada componente o_t de la secuencia es un símbolo de Σ. Durante el desarrollo de la tesis doctoral ha sido necesario resolver cada uno de los problemas básicos de los HMMs (véase sección 3.3) para conseguir los siguientes objetivos:

1. Llevar a cabo un proceso de entrenamiento *off-line* de algunos parámetros del modelo

correspondiendo con el Problema 3

2. Encontrar la mejor secuencia de estados en relación con el Problema 2

3. Calcular las probabilidades a partir de los algoritmos Viterbi y Forward-Backward utilizados en Problema 1 y Problema 2.

Todo este proceso será detallado en las siguientes secciones de esta tesis doctoral.

6.4 Modelado de un HMM extendido

En esta sección se detalla cómo realizar el modelado de los HMMs (λ_k) necesarios para la predicción de los distintos tipos de ataque multi-paso. Para ello el *Off-line training module* debe determinar cada uno de los parámetros que componen el modelo extendido, $\lambda_k = (\Sigma, S, A, Q, \pi, H)$ de manera offline (véase Figura 6.2.1). Cada uno de los parámetros obtenidos para cada modelo son almacenados en una serie de ficheros de configuración (Figura 6.4.1). Las siguientes subsecciones se explica como obtener cada uno de estos parámetros de manera genérica para cualquier ataque a modelar.

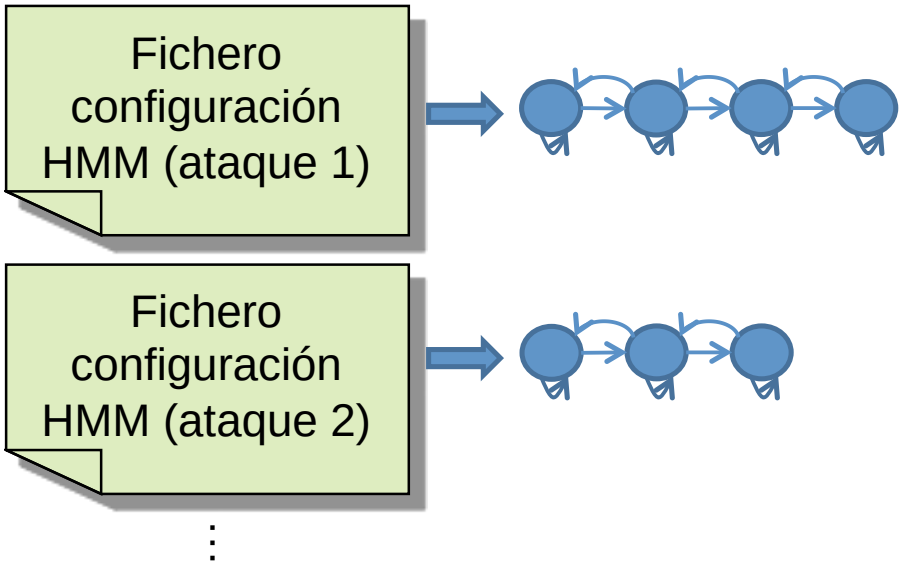


Figura 6.4.1: Ficheros de configuración de los distintos HMM modelados

6.4.1 Observaciones

Para la tesis propuesta el conjunto de observaciones, $\Sigma = \{v_1, v_2, \ldots, v_M\}$, debe representar todas las posibles alertas que un IDS puede reportar. Una manera sencilla sería tener una observación por cada tipo de alerta, pero implicaría el manejo de un conjunto demasiado grande por parte del modelo con lo que podría afectar al rendimiento del sistema de predicción. Además, cada IDS tiene sus propios tipos de alerta, por lo que hay que tener en cuenta el uso de un factor común en todos ellos para no depender de un tipo de IDS concreto. Para solventarlo, se propone la clusterización de todas las alertas basado en el reporte de CVE como contribución de la tesis doctoral.

Concretamente, cada uno de los nombres de CVE son asociados a una etiqueta basada en la frecuencia de ocurrencia de diferentes palabras en el campo de *descripcción* de cada uno de los CVEs reportados.

Además, usando estas etiquetas evitamos que se produzca sobreajuste en los modelos entrenados debido a que las observaciones engloban un conjunto de alertas posibles para la realización de los distintos pasos del atacante, asegurando así cubrir todas las posibles alertas asociadas a cada fase de ataque incluso cuando el modelo es entrenado con pocos escenarios de ejemplo.

Las observaciones obtenidas del proceso serán compartidas para todos los modelos (λ_k) de los distintos escenarios de ataque, pero si se considerara necesario se podrían tener distintos conjuntos de observaciones para los distintos tipos de ataque multi-paso.

6.4.2 Estados de la cadena de Markov

Distintas cadenas de Markov son necesarias para representar los estados de cada tipo de ataque multi-paso. Estas cadenas son construidas seleccionando los estados correspondientes a fases similares llevadas a cabo por el atacante. Estos estados son almacenados en el conjunto S del modelo λ_k asignado al tipo de ataque que se esté modelando.

La Figura 6.4.2 muestra un ejemplo con tres ataque multi-paso, cuyos ficheros de configuración han sido asignados como *ataque A*, *ataque B* e *ataque N*, con cuatro, tres y cinco

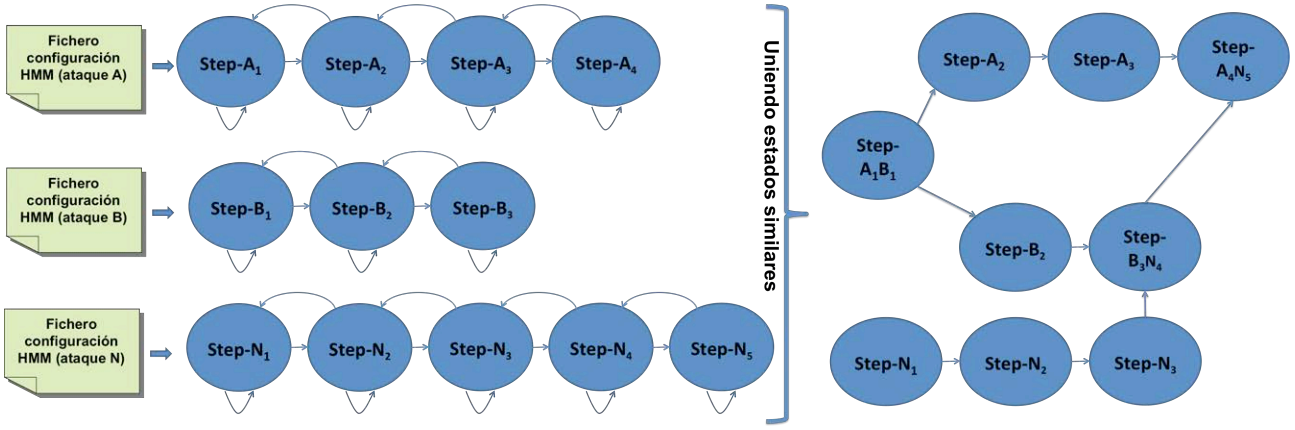


Figura 6.4.2: Ejemplo de grafo de ataque formado por distintos modelos de HMM

estados respectivamente. Además cada ataque multi-paso tiene un modelo HMM diferente que se corresponde con λ_A, λ_B and λ_C respectivamente. En el lado derecho de dicha figura se muestra como todas las cadenas de Markov pueden ser unificadas para formar un grafo de ataque, el cual puede ser utilizado por ejemplo para la visualización por parte del administrador. En este ejemplo se puede ver que hay varios estados iguales que han sido unidos como uno solo en el grafo: $Step - A_1$ y $Step - B_1$, $Step - B_3$ y $Step - N_4$, y $Step - A_4$ y $Step - N_5$.

6.4.3 Entrenamiento de matrices y vectores de probabilidad

Para conseguir la predicción de ataques es necesario una fase de entrenamiento previa basada en las observaciones obtenidas a partir de los ejemplares de alertas de distintos escenarios de ataque, es decir, se entrenan las matrices y vectores de diferentes modelos (λ_k) considerando los *Multistep attack files* que pertenecen a un mismo tipo de ataque. En concreto durante la tesis doctoral se han analizado distintos modos de entrenamiento, supervisado y no supervisado, para la validación del modelo. De esta forma se podrá determinar el mejor método de entrenamiento para la predicción de ataques multi-paso. Tras este proceso se obtienen las probabilidades asociadas a la matriz de transición de estados (A), a la matriz de probabilidad de observación (Q), al vector de distribución inicial (π) y el vector de número medio de observaciones (H); para cada uno de los modelos en estudio.

Basado en el estudio del estado del arte sobre algoritmos de entrenamiento (véase 3.3.3)

se utiliza el algoritmo Baum-Welch para obtener las probabilidades de A, Q y π en modo de entrenamiento no supervisado y en modo supervisado se entrenan A y Q mediante el uso de las fórmulas estadísticas (3.34) y (3.35) respectivamente, a partir de ejemplares disponibles de las distintas fases de los escenarios de ataque a modelar.

Finalmente, el vector de número medio de observaciones (H) se entrena estadísticamente en modo supervisado basado en el número medio de alertas con CVE de cada estado usando los *multistep attack datasets* de cada uno de los tipos de ataque.

6.5 Método de predicción de un ataque multi-paso

Una vez que todos los modelos HMM han sido entrenados con los distintos escenarios de ataque, el sistema debe predecir los posibles ataques multi-paso de la red donde se integre el sistema predictivo con el objetivo de llevar a cabo respuestas proactivas. Para ello es necesario obtener una serie de medidas o valores predictivos, de manera que puedan ser integrados en otros sistemas automáticos que realicen un procesado proactivo de dicha predicción.

Por tanto, el objetivo de la tesis doctoral es calcular la probabilidad de que se produzca un ataque multi-paso cada vez que llegan alertas desde los IDSs. Para determinar dicha probabilidad se ha optado por determinar la fase en la que se encuentra el ataque y la posibilidad de llegar a estar en el último estado de la cadena. De esta forma es posible reaccionar antes de que el sistema atacado se vea comprometido. En las siguientes subsecciones se indican los cálculos que realiza el método de predicción propuesto.

6.5.1 Secuencia y probabilidad de los estados

El Algoritmo de Viterbi se utiliza para calcular la mejor ruta de estados cada vez que hay una nueva observación obtenida desde las alertas de los IDSs. En concreto, cada vez que llega una nueva observación al sistema de predicción, se aplica el algoritmo de Viterbi sobre los ficheros de configuración de cada uno de los distintos modelos de ataque previamente obtenidos de la fase de entrenamiento offline. De este modo, es posible determinar el *tipo de ataque* que está sucediendo, *el patrón de ataque* (λ_k) y en que *fase del ataque* se encuen-

tra dentro de los estados pertenecientes al conjunto S de dicho modelo. Además, es posible determinar la probabilidad de estar en cada uno de los estados de la cadena, es decir, la probabilidad de estar en el estado s_i en el tiempo t, dada una secuencia de observación O y el modelo λ_k (3.29). Esta probabilidad puede ser expresada en términos de las variables *forward* y *backward* [YK03]:

$$\gamma_t(i) = \frac{a_t(i) \cdot \beta_t(i)}{P(O|\lambda_k)} = \frac{a_t(i) \cdot \beta_t(i)}{\sum_{i=0}^{N} a_t(i) \cdot \beta_t(i)} \tag{6.1}$$

Donde $a_t(i)$ explica la secuencia de observación parcial $\{O_1, O_2, \ldots, O_t\}$ para el estado s_i en tiempo t y $\beta_t(i)$ explica el resto de la secuencia de observación $\{O_{(t+1)}, O_{(t+2)}, \ldots, O_T\}$.

Considerando una secuencia de estados fijada $X = \{x_1, x_2, \ldots, x_T\}$, el Algoritmo Forward-Backward determina el cálculo de $a_t(i)$ y $\beta_t(i)$ con las ecuaciones (3.22) y (3.23) respectivamente.

6.5.2 Probabilidad de ataque multi-paso

Una vez calculada la probabilidad de los estados basado en la secuencia de observación del ataque en progreso, es posible predecir la probabilidad de que el atacante consiga materializar completamente el ataque multi-paso, es decir, que llegue al final de la cadena de Markov de dicho modelo de ataque. Como contribución de la tesis doctoral se ha propuesto un método para calcular la probabilidad del ataque multi-paso $P_{intrusion}$, basado en el número de observaciones obtenidas desde alertas de los IDSs y la secuencia de estados predecida por el sistema, $X = \{x_1, x_2, \ldots, x_T\}$ (véase la sección 6.5.1) del ataque en progreso de la siguiente forma:

$$P_{intrusion} = \frac{\sum_{t=0}^{E} \max_i [P(x_t = s_i|O, \lambda_k)]}{E} \tag{6.2}$$

Donde E es calculado usando la función $f(e_j)$:

$$f(e_j) = \begin{cases} h_j & \text{if } j \geq T \text{ and } h_j > c_j \\ c_j & \text{else} \end{cases} \tag{6.3}$$

$$E = \sum_{j=0}^{N} f(e_j) \tag{6.4}$$

Siendo h_j el número medio de observaciones para el estado j y c_j el contador del número de observaciones del ataque en progreso detectado en cada estado ($C = \{c_1, c_2, \ldots, c_T\}$). Los valores del contador son ajustados dinámicamente en función del número de observaciones del ataque actual. La condición $h_j > c_j$ es usada como factor de corrección para volver a estados previos, debido por ejemplo a la realización de respuestas proactivas.

6.6 Validación del HMM extendido

La predicción de ataques multi-paso es esencial para las organizaciones debido al incremento de ciberataques. La Denegación de Servicio Distribuida (DDoS) es uno de estos tipos de ataque más preocupantes a día de hoy con numerosos ejemplos en diversas organizaciones [Cer14] [Dyn16] como en plataformas de juego on-line (Pokemon Go), en compañías de hosting en internet (OVH), y proveedores de servicios de DNS (Dyn). Por este motivo, la validación del modelo predictivo de ataques multi-paso se va a centrar en este tipo de ataque.

6.6.1 Modelado de DDoS mediante un HMM extendido

En esta subsección se describe el modelo de HMM extendido realizado para los escenarios de ataque de DDoS. A continuación se detallan los distintos parámetros del modelo $\lambda_{ddos} = (\Sigma, S, A, Q, \pi, H)$.

6.6.1.1 Observaciones

Como se mencionó en la sección 6.4.1, las observaciones se obtienen de un proceso de clusterización de las alertas tras un previo análisis del campo de *descricción* de los CVE reporta-

dos. Concretamente el proceso a realizar es el siguiente:

1. El informe de CVE es descargado en formato XML (eXtensible Markup Language) desde la página web oficial [cve].

2. Se aplica un parser desarrollado para este tesis doctoral al fichero XML usando la API SAX de Java para extraer y almacenar todos los nombres de CVE y sus descripciones.

3. Todas las palabras de dichas descripciones son contabilizadas automáticamente usando una implementación desarrollada en C#. Como resultado se obtiene una lista de palabras y su número de ocurrencias para todas las descripciones de los CVEs. De todas ellas se seleccionan las palabras con mayor ocurrencia, de tal manera que, todos los CVEs tiene asociado como mínimo una palabra

4. Finalmente, estas palabras seleccionadas son agregadas en términos relacionados con la ciberseguridad de tal manera que las etiquetas pueden estar compuestas por varias palabras.

Los resultados de este proceso de clusterización fueron: *Denial of Service*, *Buffer Overflow*, *Cross-site Scripting*, *Remote Attackers*, *Remote Users*, and *Information*. Las etiquetas *Remote Attackers* y *Remote Users* parecen representan un concepto similar, sin embargo, para el reporte de CVE el segundo término se considera como usuarios válidos que acceden al sistema de manera remota.

Además, a cada una de estas etiquetas se les asigna un valor como indicativo del grado de especificidad; de esta forma si un CVE contiene más de una etiqueta en su campo de descripción, solo es asociado con la etiqueta más específica. De las etiquetas obtenidas, *Information*, *Remote Attacker*, y *Remote User* son considerados menos específicos que *Cross-site Scripting*, *Buffer Overflow*; los cuales a su vez son menos específicos que la etiqueta *Denial of Service*. Esta información es almacenada en la *Security database* (véase Figura 6.2.1).

La Figura 6.6.1 muestra el esquema de todo el proceso de clusterización propuesto en la tesis doctoral para la correspondencia de alertas basado en su CVE. La *Security DB* almacena

estas etiquetas de manera persistente en dos tablas; una de ellas contiene todas las etiquetas obtenidas tras la clusterización del informe de CVE (Figura 6.6.1) y la segunda tabla asocia cada CVE con su correspondiente etiqueta, como se puede ver en el ejemplo de la Figura 6.6.2.

Finalmente el conjunto de observaciones Σ se basa en estas etiquetas y en la posible severidad de las alertas (*info*, *low*, *medium*, y *high*). En concreto, las alertas con severidad *info* son descartadas ya que solo proporciona avisos.

Una vez que tenemos el conjunto de observaciones del modelo, tanto en el proceso de entrenamiento como para el proceso predictivo de ataques, las observaciones son obtenidas de las alertas siguiendo el esquema del ejemplo de la Figura 6.6.2. Cuando llega una alerta desde los IDSs o desde el fichero de escenarios de ataque es convertido a formato IDMEF. Tras ello, se obtiene el nombre de CVE siguiendo la etiqueta *Reference* con valor de *origin=çve"* . Este nombre de CVE es buscado en la base de datos para obtener la etiqueta a la que corresponde la alerta. Finalmente de la etiqueta *Impact* en la propiedad *severity* se encuentra el valor de severidad de la alerta. Uniendo estos dos valores se obtiene la observación.

6.6.1.2 Estados de la cadena para el ataque de DDoS

La selección de estados de la cadena de Markov para un ataque de DDoS se ha basado en los fases propuestas en el escenario de ataque *LLDDOS1.0* de DARPA (Defense Advanced Research Projects Agency) del año 2000 [DAR00]. Las cinco fases son:

1. Barrido de IPs de la red desde un sitio remoto.
2. Escaneo de IPs para buscar el demonio sadmind ejecutado en máquinas Solaris.
3. Explotación de una vulnerabilidad del demonio sadmind, intentando tanto en todas las máquinas se tenga o no éxito.
4. Instalación del troyano, mstream DDoS software, sobre tres máquinas de AFB.

Paso	Nombre	Tiempo	Objetivo
1	*IPsweep*	15:51:36.142272 - 15:52:00.818096	El atacante envía peticiones ICMP echo-requests en barrido y queda a la espera de ICMP echo-replies para determinar los host activos.
2	*Sadmind ping*	16:08:07.354091 - 16:18:05.548183	Los hosts descubiertos son probados para determinar cuales ejecutan la herramienta de administración remota *sadmind*. ESto indica a los atacantes que hosts podrían ser vulnerables a ser explotados.
3	*Break into*	16:33:10.611612 - 16:35:01.720586	Los atacantes intentan acceder a los hosts descubiertos utilizando la vulnerabilidad de *sadmind*.
4	*Installation*	16:50:01.749301- 16:50:54.049012	Instalación del troyano *mstream DDoS software* sobre estos hosts.
5	*Launch*	17:26:15.437497	Ejecución del DDoS.

Tabla 6.6.1: Los cinco pasos del escenario de ataque LLDOS1.0 de DARPA

Nombre	Alertas Snort	CVE	Severidad
IPsweep	Protocol-icmp ping	No tiene CVE	-
Sadmind ping	Protocol-RCP Solaris UDP portamap sadmin port query request attempt	CVE 2003-0722	Medium
	Protocol-RPC portmap sadmind request UDP attempt	CVE 1999-0977	Low
	Protocol-RPC sadmind UDP ping	No CVE name	-
Break into	Protocol-RPC sadmind UDP netmgt_proc_service client-domain overflow attempt	CVE 1999-0977	High
Installation	Protocol-services rsh root	No tiene CVE	-
Launch	Protocol-SNMP request UDP	CVE 2002-0013	Medium
		CVE 2002-0012	Medium

Tabla 6.6.2: Alertas Snort y nombre de CVE reportado en cada paso del escenario de ataque LLDOS1.0 de DARPA

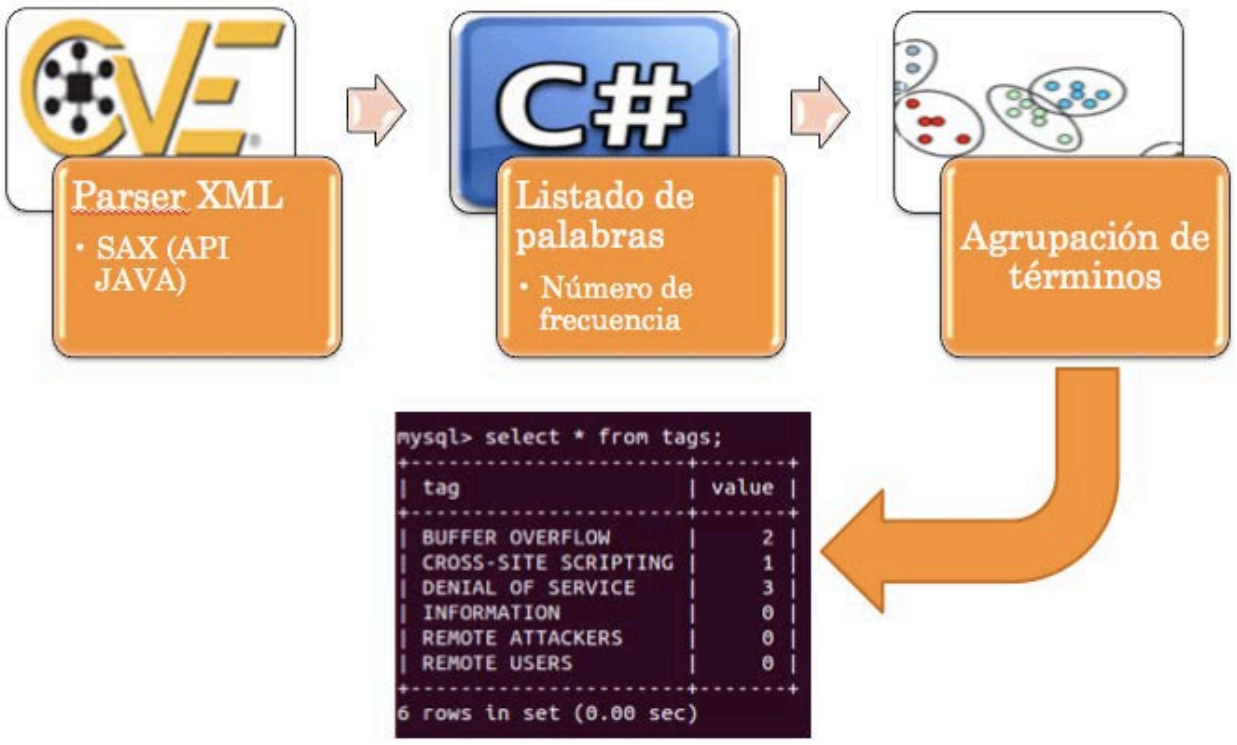


Figura 6.6.1: Proceso de clusterización de vulnerabilidades basado en el CVE de las alertas

5. Ejecución del DDoS.

Estas fases de ataque están descritas en mayor detalle en la Tabla 6.6.1. El tiempo de cada uno de los pasos ha sido obtenido usando la aplicación Wireshark a partir de los ficheros *pcap* de cada fase. El objetivo de cada fase han sido obtenidos de [SDJC12].

Para el caso de estudio del sistema de predicción propuesto en la tesis doctoral se ha utilizado Snort [sno] como IDS de red. Otra consideración a tener en cuenta es que la solución propuesta se basa en el reporte de vulnerabilidades CVE, por lo que sólo podremos considerar alertas que estén contenidas en este reporte. La Tabla 6.6.2 muestra algunos ejemplos de alertas representativas de cada fase reportadas por Snort para las trazas *pcap* del escenario LLDDOS1.0. Dado nuestro enfoque, el campo CVE es obligatorio por lo que solo se pueden considerar tres de las cinco fases del escenario LLDDOS1.0 (paso 2: Sadmind ping, paso 3: Break into, y paso 5: Launch). Para un ataque de DDoS genérico estas fases pueden considerarse como *intrusion attempt*, *compromised system*, y *denial of service*, respectivamente para los pasos 2, 3 y 5 de este escenario. Otra consideración a tener en cuenta es que para

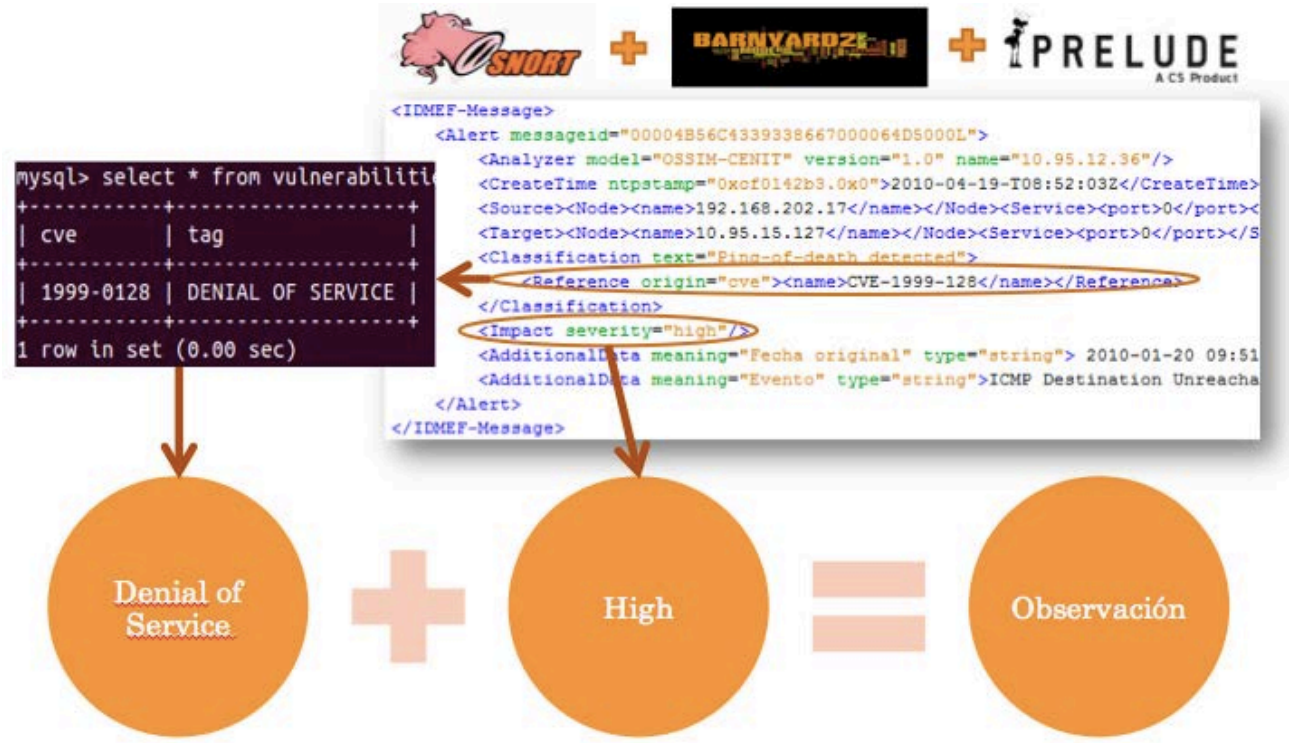


Figura 6.6.2: Ejemplo de observación asociada a una alerta IDMEF

este caso de estudio solo se va a entrenar con ataques de *Buffer Overflow* para ganar acceso a sistemas remotos; por este motivo el modelo de Markov exclusivamente predice este tipo de vulnerabilidad en la fase de *compromised system*. Como resultado, los estados de la cadena de Markov se ven representados en la Figura 6.6.3 (El modelo es completamente conectado, por visibilidad no se muestran todas las posibles transiciones entre estados).

6.6.1.3 Entrenamiento de matrices y vectores de probabilidad

Para el entrenamiento del caso de estudio se han analizado dos escenarios de ataque disponibles en DARPA 2000 debido a que son los únicos datasets con trazas separadas en cada una de las fases del ataque multi-paso. La necesidad de tener previamente las trazas separadas por fases es debido al modo de entrenamiento supervisado, ya que sin ellas solo sería posible la realización del caso de estudio con entrenamiento en modo no supervisado. Además facilita la validación del modelo propuesto.

Por un lado, se ha seleccionado el algoritmo de Baum-Welch con escalado para el entre-

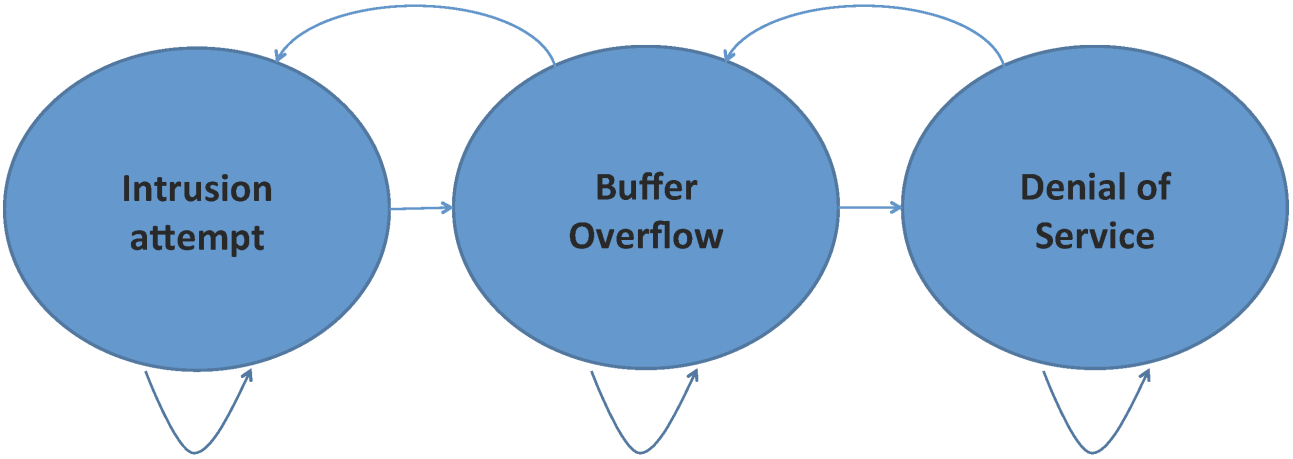


Figura 6.6.3: Estados de la cadena de Markov para ataques de DDoS

namiento en modo no supervisado. La implementación de dicho algoritmo se ha llevado a cabo en Java usando la librería jahmm. Por otro lado, el algoritmo supervisado no estaba incluida en la librería por lo que fue implementada directamente en Java. Ambos métodos han sido explicados en la sección 6.4.3.

A continuación se indican los valores de las matrices y vectores del modelo HMM tras el entrenamiento de la cadena propuesta en la Figura 6.6.3 tanto con el método supervisado como el no supervisado. En concreto se ajustan los valores de A, Q, and π que maximizan la probabilidad de de las secuencias de los escenario de prueba usando ambos métodos y el vector H obtenido solo en modo supervisado.

Entrenamiento no supervisado Para la realización del entrenamiento con el algoritmo Baum-Welch es necesario crear previamente un modelo λ_{ddos} con valores de probabilidad aleatorias para después aplicar iterativamente el algoritmo. Este aprendizaje puede aplicarse en secuencias de observación largas sin generar *underflows*. Además, se aplicó un método que mide la distancia entre dos HMM como condición de parada del proceso de entrenamiento; en concreto se utilizó la distancia de Kullback-Leibler [Rab85]. De este modo, el proceso de entrenamiento se detiene cuando no se producen mejoras apreciables en el HMM y por tanto, el modelo converge.El algoritmo se ejecutó múltiples veces cambiando tanto el número de iteraciones como la distancia aceptable de mejora obteniendo resultados similares,

Observaciones		Estados					
		Intrusion attempt		Buffer overflow		Denial of service	
Etiqueta	**Severidad**	Supervisado	No supervisado	Supervisado	No supervisado	Supervisado	No supervisado
Information	Medium	0	0	0	0,011	0,011	0,011
Remote users	Medium	0	0	0	0,065	0,064	0,065
Remote attackers	Medium	0,962	0,832	0,667	0	0	0
	High	0	0	0	0,379	0,372	0,379
Buffer overflow	Low	0,038	0,022	0	0	0	0
	High	0	0,139	0,333	0,013	0,021	0,013
Denial of service	Low	0	0	0	0,022	0,021	0,022
	Medium	0	0	0	0,498	0,489	0,498
	High	0	0,006	0	0,012	0,021	0,012

Tabla 6.6.3: Vectores de observación con entrenamiento supervisado y no supervisado

por lo que el modelo convergía en el mismo punto. De este modo se obtuvo el vector de distribución inicial, la matriz de probabilidad de transición y la matriz de probabilidad de observación.

- El vector de distribución inicial (6.5) indica el estado inicial del ataque. En este caso, se puede ver que en el 100 % de los casos el primer estado (*intrusion attempt*) es el estado inicial.

$$\pi = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \tag{6.5}$$

- La matriz de probabilidad de transición (6.6) que describe las transiciones entre estados del conjunto S, en concreto los que componen la cadena de la Figura 6.6.3. Por ejemplo, la probabilidad de estar en el estado *intrusion attempt* es 0,985.

$$A = \begin{pmatrix} 0,985 & 0,007 & 0,007 \\ 0 & 0,5 & 0,5 \\ 0 & 0,5 & 0,5 \end{pmatrix} \tag{6.6}$$

- La matriz de probabilidad de observación es estructurada por un vector de probabilidad para cada estado. En la Tabla 6.6.3 se representa cada vector teniendo en cuenta las observaciones con valores relevantes.

Entrenamiento supervisado El entrenamiento supervisado aplicado utiliza métodos estadísticos para obtener el vector medio de observaciones y las probabilidades de distirbución inicial, transición y observación. Los resultados son los siguientes:

- El vector de distribución inicial se corresponde con (6.7). El primer estado podría ser *intrusion attempt* con una probabilidad de 0,348.

$$\pi = \begin{pmatrix} 0,348 & 0,238 & 0,414 \end{pmatrix} \tag{6.7}$$

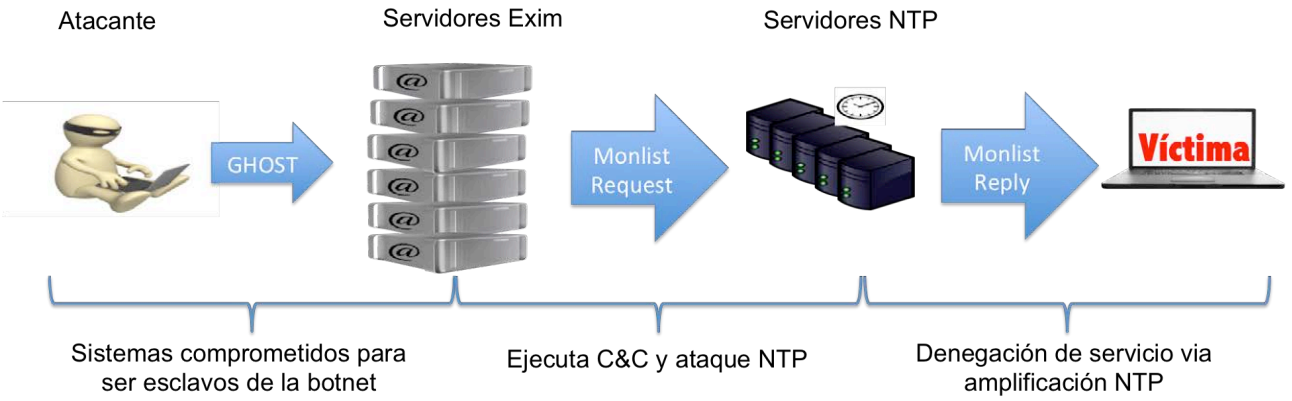


Figura 6.6.4: Escenario de prueba de ataque de DDoS

- La transición entre estados de la cadena respresentada en Figura 6.6.3 se corresponde con la matriz (6.8). Por ejemplo, la probabilidad de permanecer en el estado de *denial of service* es de 0,979. Este valor es normal porque el aprendizaje del modelo solo está utilizando los paso de ataque sin considerar las posibles respuestas.

$$A = \begin{pmatrix} 0,974 & 0,013 & 0,013 \\ 0,010 & 0,962 & 0,028 \\ 0,011 & 0,011 & 0,978 \end{pmatrix} \tag{6.8}$$

- En la Tabla 6.6.3 se muestra los vectores de probabilidad de observación para cada estado del modelo de DDoS propuesto, teniendo en cuenta solo las observaciones con valores relevantes.

- Los valores de número medio de observaciones en los escenarios de DDoS usados en el entrenamiento son los siguientes:

$$H = \begin{pmatrix} 79 & 27 & 47 \end{pmatrix} \tag{6.9}$$

6.6.2 Escenario de prueba de ataque de DDoS

Como contribución de la tesis doctoral se ha creado y desplegado el escenario virtual mostrado en la Figura 6.6.4 para evaluar el modelo propuesto (λ_{ddos}) con ataques más actuales y desconocidos para dicho modelo. A continuación se explican los pasos específicos del escenario de prueba propuesto:

1. Creación de una botnet mediante un ataque de *Buffer Overflow* para acceder a varios servidores de correo. Este ataque podría ser reemplazado por cualquier otro ataque de acceso remoto, por ejemplo explotando la vulnerabilidad Shellshock, pero el modelo está entrenado solo para ataques de *Buffer Overflow*.

2. Una vez que todos los servidores de correo han sido comprometidos, se ejecuta un script malicioso. Este script lanza peticiones a varios servidores de NTP con el objetivo de que este servidor responda con una respuesta grande hacia la máquina víctima. El tráfico generado por todos estos servidores hacia la misma máquina originan una DDoS. El modelo de Markov propuesto es capaz de detectar cualquier DoS registrado en el reporte de CVE, por lo que aunque no se haya entrenado con ningún patrón de este ataque, debe ser capaz de predecirlo.

Este ataque multi-paso ha sido desplegado en un escenario virtual VNX (Virtual Networks over linuX) [vnx]. El DDoS simulado incluye la fase 2 y 3 de la cadena de Markov (Figura 6.6.3) como se describe en los siguientes subapartados.

6.6.2.1 Buffer Overflow

El objetivo del atacante es ejecutar código de manera remota. En este caso, el atacante se aprovecha de la vulnerabilidad GHOST (CVE 2015-0235) con este objetivo. GHOST es considerado una vulnerabilidad crítica debido a que los atacantes podrían utilizarlo para obtener silenciosamente el control completo de un sistema Linux sin tener ningún conocimiento previo de las credenciales del sistema.

La vulnerabilidad GHOST puede ser explotada tanto local como remotamente utilizando las funciones *gethostbyname*()* de la librería *glibc*, la cual forma parte del núcleo del sistema operativo Linux. Este problema se origina en un *Buffer Overflow* basado en *heap* encontrada en la función *_nss_hostname_digits_dots()* de la librería *glibc*. Esta función se invoca especialmente en las llamadas a las funciones *_gethostbyname* and *gethostbyname2()*. Un atacante remoto capaz de hacer una llamada a la aplicación con cualquiera de estas funciones podría utilizar este agujero de seguridad para ejecutar código arbitrario con los permisos del usuario que ejecuta dicha la aplicación.

La primera versión de la libreria GNU C afectada por esta vulnerabilidad es glibc-2.2 del 10 de Noviembre del año 2000. El error fue solucionado el 21 de Mayo de 2013 (entre las versiones de glibc-2.17 y glibc-2.18). Desafortunadamente, no fue reconocido como una amenaza de seguridad y como resultado, posteriormente hubieron distribuciones estables y de largo plazo que estuvieron expuestas, incluyendo Debian 7 (wheezy), Red Hat Enterprise Linux 6 & 7, CentOS 6 & 7, and Ubuntu 12.04.

Hay software que inicia conexiones de red, procesamiento de registros o filtrado de correo / spam usando *gethostbyname()*. En concreto para este escenario de prueba se ha utilizado máquinas Ubuntu 12.04 vulnerables con la versión EGLIBC (Embedded GLIBC) 2.15-oubuntu10.3 y el servidor de correo Exim 4.77. En cuanto al atacante, es una máquina Kali 1.1.0 con un exploit de ghost adicional que se incluyó a *Metasploit Framework*. El código del exploit GHOST fue implementado por Qualys, Inc. [qua].

6.6.2.2 Denial of service

El ataque de amplificación de NTP es utilizado para la realización de DDoS. El uso de la fución de supervisión de servidores NTP (*monlist*) con versiones anteriores a 4.2.7p26 o mal configuradas, provoca el efecto de amplificación de la respuesta. La consulta *monlist* obtiene la lista de las máquinas más recientes que han interactuado con el servidor. Obviamente, esta respuesta puede ser bastante grande en relación con la consulta. Esta funcionalidad ha sido considerada como una vulnerabilidad del protocolo y asociada al CVE-2013-5211 [Cer14]. Dos fases componen este ataque:

IP spoofing La comunicación entre un cliente y un servidor NTP está basado en UDP (User Datagram Protocol). Las características del protocolo UDP permite la modificación de las direcciones IP en un flujo de tráfico y mantener su validez. El atacante construye consultas NTP y reemplaza la dirección de IP origen en los datagramas con la IP de la máquina objetivo. De esta forma, todas las respuestas van a la víctima (lo que se denomina reflexión).

Monlist query Sobre servidores NTP vulnerables es posible conseguir una respuesta mucho más grande que la consulta (lo que se denomina amplificación de la respuesta). Además, varias consultas *monlist* son lanzadas a todos los servidores NTP del escenario virtual para conseguir un mayor volumen de tráfico hacia la víctima.

6.6.3 Evaluación del sistema de predicción para DDoS

La validación del sistema consta de varias partes. En el primer caso se analizan los dos modos de entrenamiento usando la secuencia de estados obtenida tras aplicar el sistema de predicción propuesto. En segundo lugar, se valida el diseño propuesto con alertas incluidas en el *Multistep attack files* y por tanto conocidas para el sistema. Por otro lado, se verifica la efectividad del modelo de predicción propuesto frente a alertas desconocidas para el sistema de predicción utilizando el escenario de prueba propuesto en la sección 6.6.2. Finalmente, se presentan medidas de rendimiento del sistema propuesto, demostrando que el modelo es aplicable en tiempo real.

Para todas las pruebas se han utilizado alertas en formato IDMEF obtenidas desde el IDS Snort aplicando la herramienta Barnyard2. Todos los resultados han sido obtenidos localmente en una máquina Ubuntu 14.04 equipada con un procesador Intel Quad Core i5-3470 a 3,20 GHz, con 16 GB RAM y 500 GB de memoria disco.

Validación de modos de entrenamiento Como se explicó en la sección 6.5.1, el algoritmo de Algoritmo de Viterbi utilizando las matrices de probabilidad cada vez que llega una alerta al sistema es capaz de obtener la secuencia de estados más probable. La figura Figura 6.6.5 and

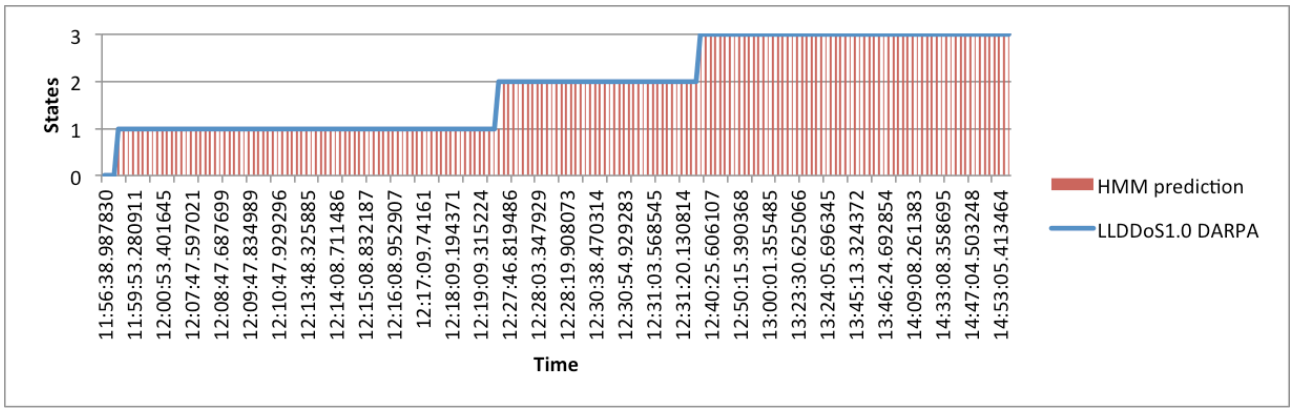


Figura 6.6.5: Secuencia de estados con aprendizaje supervisado

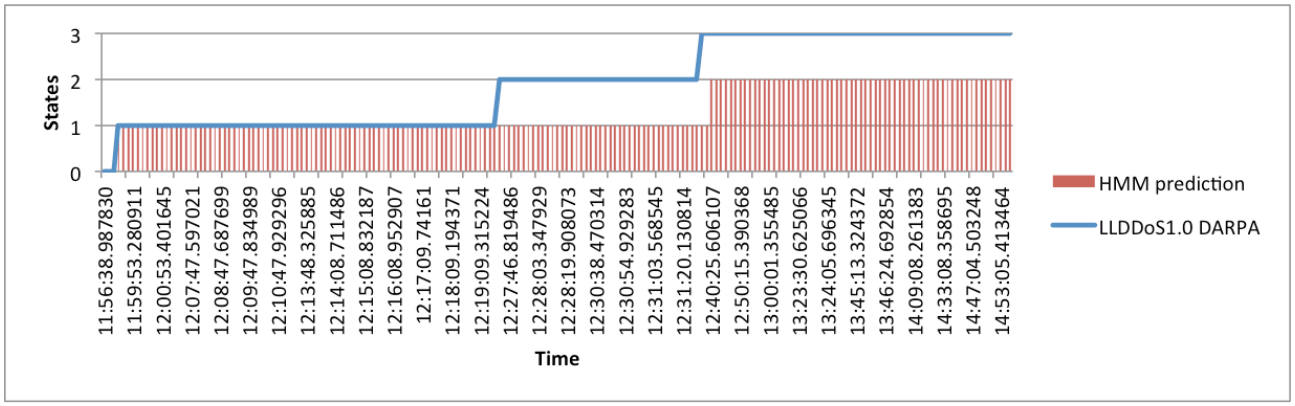


Figura 6.6.6: Secuencia de estados con aprendizaje no supervisado

Figura 6.6.6 muestran los resultados obtenidos usando el modelo λ_{ddos} (véase sección 6.6.1). En el caso de la primera figura, el modelo se ha entrenado utilizando el método supervisadoy en el segundo caso, entrenado en modo no supervisado (6.6.1.3). En ambos casos el eje OY representa el estado del DDoS genérico explicado en la sección 6.6.1.2 (*Normal*, 0; *Intrusion attempt*, 1; *Buffer Overflow*, 2; and *Denial of Service*, 3). El resultado mostrado por *LLDDoS1.0 DARPA* en ambas gráficas representa los tres estados del escenario de ataque LLDOS1.0 que contienen alguna alerta con CVE asociado (véase Tabla 6.6.2).La predicción de estados es representada por las líneas verticales, donde cada una de ellas es la predicción del sistema cuando ha llegado una nueva alerta (*HMM prediction*).

El tiempo de la Tabla 6.6.1 y el representado en el eje OX de las gráficas (Figura 6.6.5 y Figura 6.6.6) difiere debido a que el parser de Prelude es muy lento, y como resultado se ha

tenido que reenviar los paquetes del fichero *pcap* utilizando la herramienta *tcpreplay* a un ratio menor. Consecuentemente, el tiempo de detección se corresponde con el tiempo en el que se llevó a cabo la prueba, la cual comenzó aproximadamente a las 12:00.

Por un lado, como representa la Figura 6.6.5, el sistema propuesto es capaz de detectar y predecir tres fases de DDoS en el caso de entrenamiento supervisado. Además muestra que el sistema predice correctamente los estados para todas las alertas recibidas.

Por otro lado, en el caso de que el sistema se entrene en modo no supervisado (Figura 6.6.6), el sistema no es capaz de detectar las tres fases debido a que el HMM no ha sido entrenado con suficientes muestras de escenarios de ataque de DDoS.

Como conclusión, para el entrenamiento correcto de manera no supervisada es necesario un mayor número de *datasets* para el entrenamiento del sistema de predicción. Sin embargo, el sistema está preparado para detectar las distintas fases de DDoS en modo supervisado.

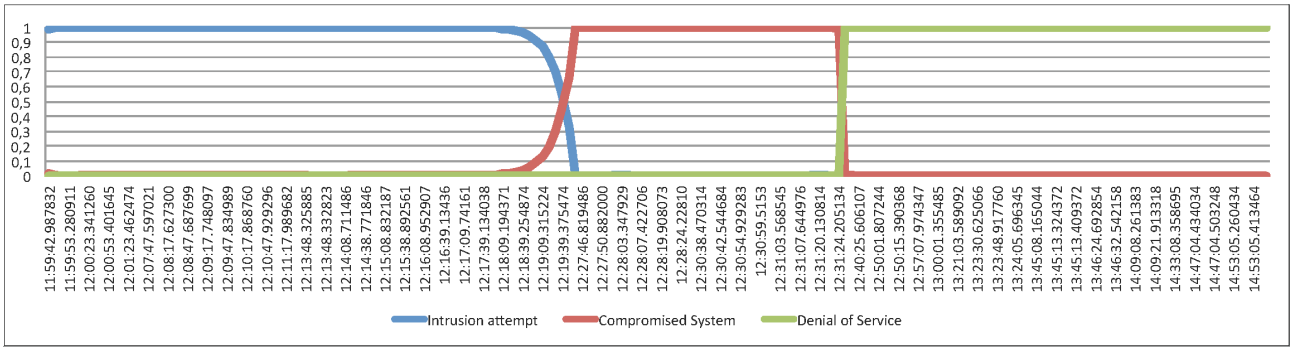


Figura 6.6.7: Probabilidad de cada estado ($\gamma_t(i)$)

Probabilidad de cada estado Como se ha comprobado en la sección 6.6.3, el sistema está preparado para predecir los estados de un ataque multi-paso utilizando las matrices de probabilidad entrenadas en modo supervisado. Cada vez que llega una alerta, el sistema calcula la probabilidad de encontrarse en cada uno de los estados de la cadena de Markov de los modelos entrenados. Este valor es necesario para posteriormente poder calcular la probabilidad de que se produzca un ataque multi-paso basado en los estados más probables de la cadena. Para validar la probabilidad calculada para cada estado se ha utilizado el escenario LLDDOS1.0 de DARPA 2000.

Los resultados de $\gamma_t(i)$ se muestran en la Figura 6.6.7. El eje Y representa la probabilidad de estar en un estado concreto de la cadena de Markov del modelo λ_{ddos} basado en las observaciones que han ido llegando al sistema de predicción. El eje X indica el tiempo de llegada de cada una de las alertas al sistema de predicción. Si comparamos estos resultados con los resultados obtenidos en la Figura 6.6.5, se observa que el sistema de predicción predice correctamente los estados del ataque con una probabilidad muy alta.

Por otro lado, es posible utilizar símplemente la variable forward para calcular la probabilidad condicional y obtener resultados similares mejorando el coste computacional como se puede ver en la Figura 6.6.7, aunque la precisión de la probabilidad calculada sea menor.

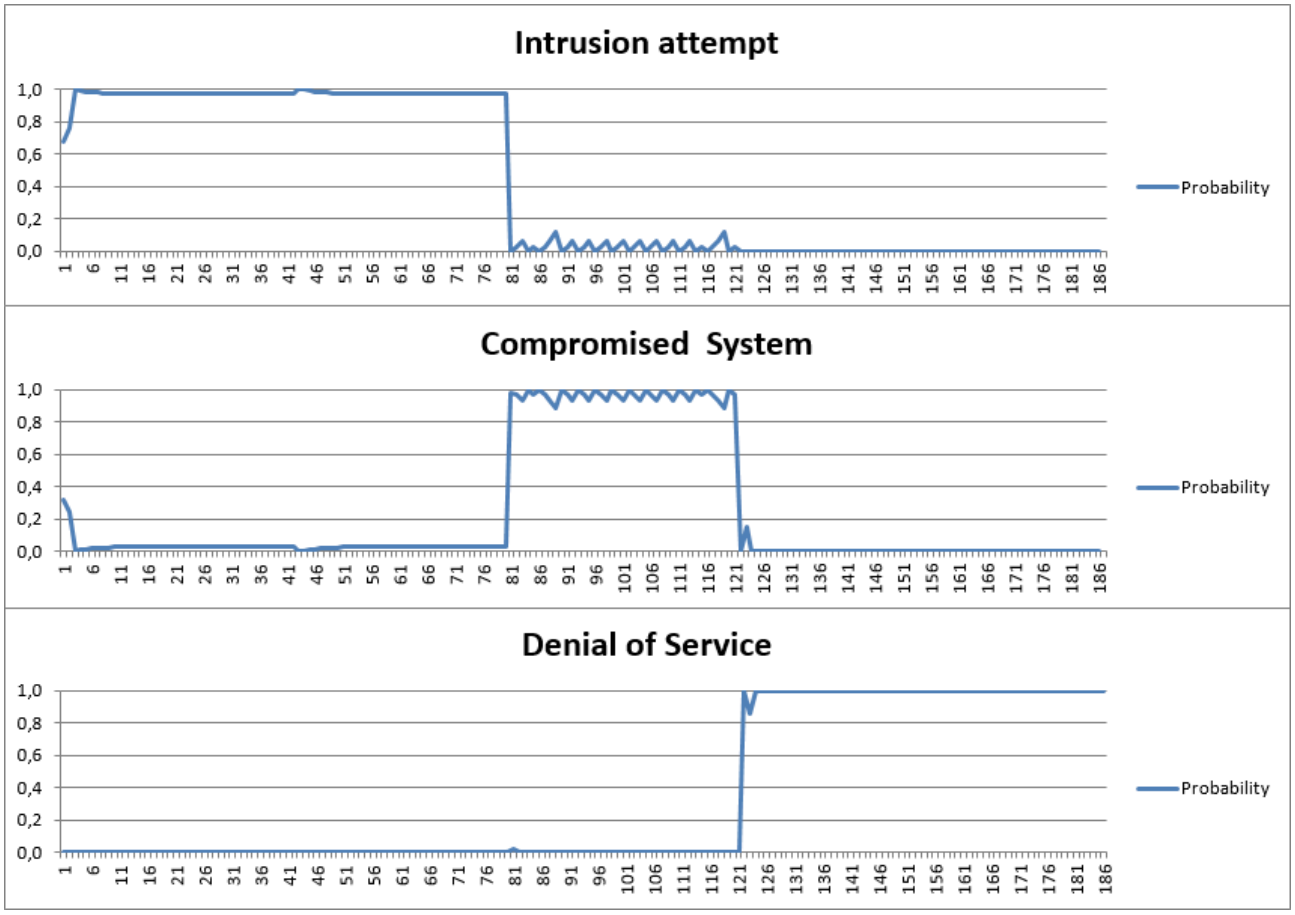


Figura 6.6.8: Probabilidad de cada estado (variable forward)

En conclusión, estos resultados indican que el modelo HMM propuesto para DDoS es capaz de obtener el estado en el que se encuentra el ataque y su probabilidad para cada ob-

servación de forma correcta.

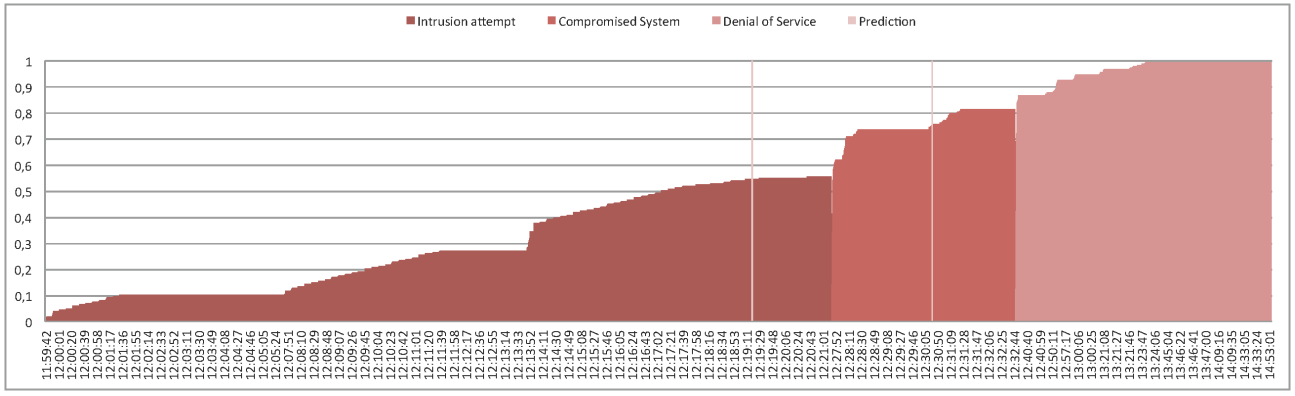


Figura 6.6.9: Probabilidad de ataque de DDoS usando las trazas de LLDDoS1.0

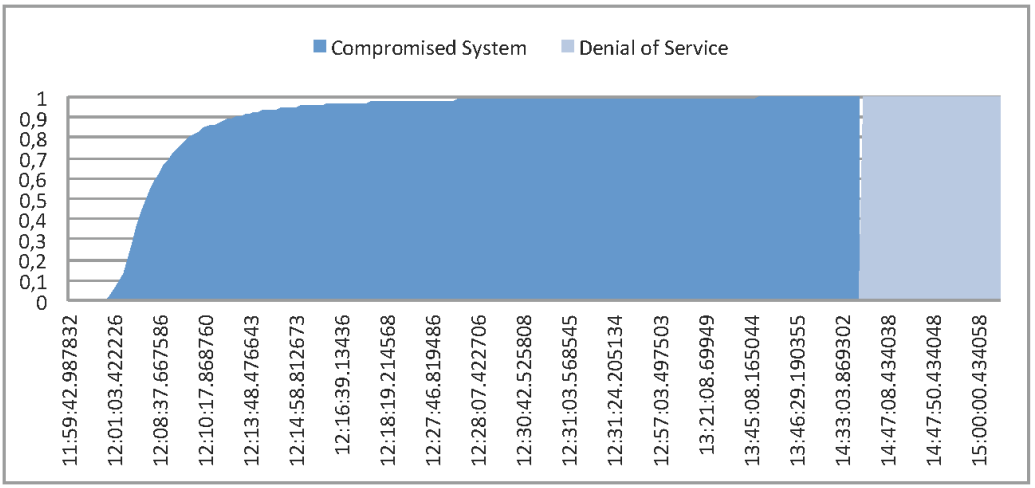


Figura 6.6.10: Probabilidad de ataque de DDoS usando el escenario virtual de ataque NTP

Probabilidad de ataque multi-paso Finalmente, se utilizan las probabilidades de cada estado, $\gamma_t(i)$, para calcular la probabilidad que se produzca un ataque multi-paso, es decir, la probabilidad de que el atacante consiga su objetivo.

La probabilidad de que se produzca un ataque multi-paso es calculada basándose en la ecuación (6.2). La Figura 6.6.9 muestra los estados y la probabilidad de que se produzca finalmente el ataque de DDoS utilizando las trazas del escenario LLDDOS1.0. Por ejemplo,

cuando el IDS envía una alerta a las 12:30:42, el sistema de predicción indica que 75 % del ataque en progreso ha sido realizado.

Estas probabilidades y fases de ataque pueden ser usadas para determinar un conjunto de políticas de respuesta proactiva dependiendo de los diferentes factores, tales como el tipo de ataque o la importancia de cada activo de la organización. Otro punto de vista, incluido en [SDJC12], especifica diferentes instantes de predicción obteniendo una probabilidad del 95 %. De modo similar, la Figura 6.6.9 muestra dos líneas verticales en los instantes de predicción del modelo HMM propuesto al 95 % de estar en cada estado, basado en la ecuación (6.2). Debe tener en cuenta que el denominador de esta ecuación se correspondería con el número medio de alertas almacenadas en H para cada estado. Concretamente, la primera línea a las 12:19:19 horas indica que el sistema predice cerca de 9 minutos antes del siguiente estado (*Buffer Overflow*) y 1 minuto 20 segundos antes de que *Intrusion attempt* finalice. La segunda línea de predicción es a las 12:30:42 horas e indica que el sistema predice 11 minutos antes de que el atacante comience el siguiente paso de DoS y cercano a 1 minuto antes de que finalice el estado de *Buffer Overflow*.

La Figura 6.6.10 muestra la probabilidad de ataque calculada para el escenario virtual representado en la sección 6.6.2. Por visibilidad solo se tiene en cuenta el tráfico generado por un servidor NTP. Con estos resultados se demuestra que el modelo HMM propuesto como contribución de la tesis doctoral es capaz de detectar los estados de un ataque desconocido para el modelo y por tanto predecir la probabilidad de que el atacante cumpla su objetivo.

Como conclusión, el sistema de predicción propuesto por esta tesis doctoral es capaz de detectar las fases de un ataque multi-paso ya sea o no conocido para el modelo entrenado.

Medidas de rendimiento del sistema de predicción La realización de todos estos cálculos en tiempo real es una condición indispensable para garantizar la predicción de ataques, de forma que sea posible anticiparse a los pasos del atacante en entornos reales. La evaluación del rendimiento del sistema propuesto como contrinución de tesis se muestra en Figura 6.6.11. Esta gráfica muestra el tiempo de procesado de la probabilidad de ataque y el tiempo ne-

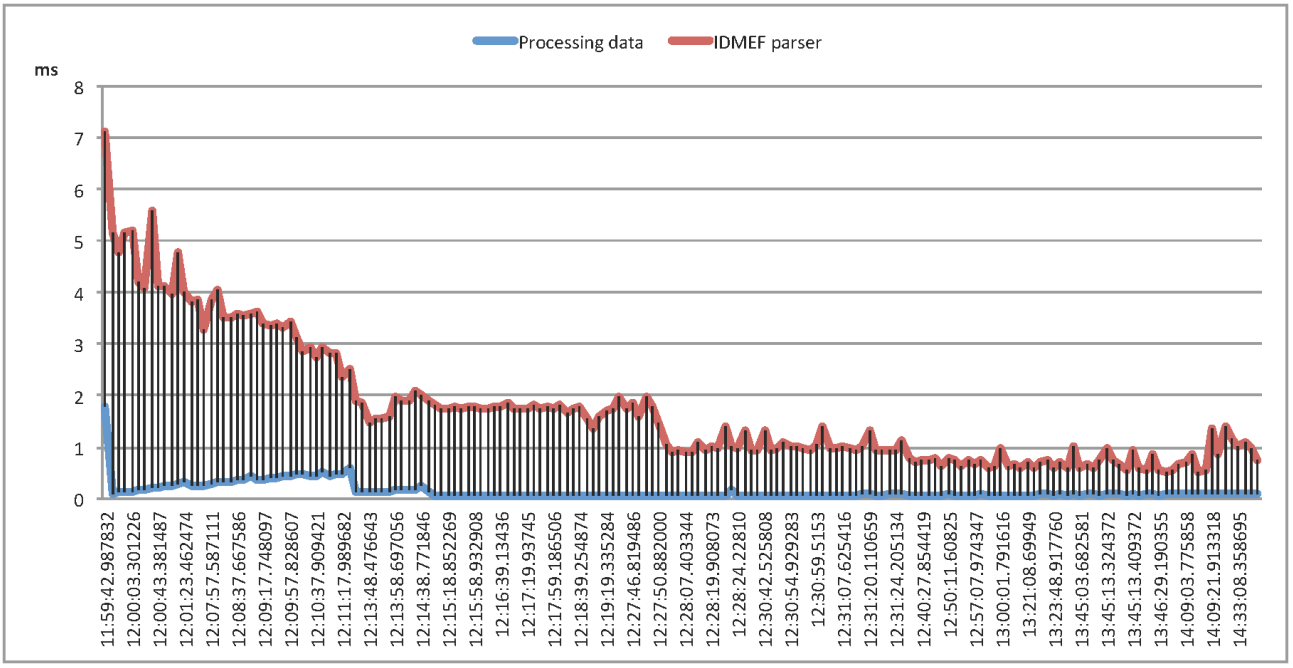


Figura 6.6.11: Medidas de rendimiento del sistema de predicción

cesario para parsear las alertas a formato IDMEF en milisegundos. El tiempo medio es de 0,14 y 1,6 milisegundos respectivamente. Además, el tiempo no incrementa con el número de observaciones por lo que la predicción de ataques multi-paso en tiempo real es viable independientemente del número de alertas reportadas por los IDSs.

Como conclusión, las medidas de rendimiento del sistema de predicción propuesto cumple con los requerimientos de un sistema en tiempo real. De este modo queda demostrado la viabilidad del sistema predictivo basado en HMM propuesto como contribución de esta tesis doctoral.

6.7 Conclusiones

En este capítulo se ha detallado la principal contribución de la tesis que trata sobre la predicción de ataques multi-paso. El diseño se ha basado en la obtención de un modelo HMM para cada escenario distinto de ataque. Todas estas cadenas conforman un grafo con todos los posibles pasos que puede llevar a cabo el atacante.

A partir de la definición formal de un HMM, se ha propuesto una extensión para poder

llevar a cabo la probabilidad de que el atacante consiga su objetivo final. Además, las observaciones necesarias para el modelo se obtiene mediante la clusterización del reporte de vulnerabilidades. De este modo, obtenemos un modelo generalizado de HMM, evitando así el problema de sobreajuste que el Aprendizaje Automático mediante HMM tiene por defecto.

La validación del sistema predictivo se ha realizado utilizando como caso de estudio el ataque de DDoS. En concreto se valida tanto la capacidad predictiva de ataques conocidos como desconocidos para el modelo HMM a través de trazas obtenidas de un escenario de ataque y finalmente se verifica el rendimiento del sistema en tiempo real.

A partir de un entrenamiento previo de todos los parámetros de la definción formal extendida se obtienen los valores de probabilidad necesarios para la predicción en tiempo real. Los resultados de validación se han basado en ataques de DDoS, obteniendo resultados prometedores.

7

CONCLUSIONES Y TRABAJOS FUTUROS

7.1 Introducción

Durante este capítulo se detalla el análisis de las dos contribuciones de la tesis doctoral expuestas en el capítulo 5 para el cálculo de la eficiencia de la respuesta y en el capítulo 6 para la predicción de ataques multi-paso. Posteriormente se comentan posibles líneas futuras de investigación de dichas contribuciones.

Finalmente se indica la difusión de resultados obtenidos durante el desarrollo de la tesis doctoral.

7.2 Análisis de las contribuciones desarrolladas en la tesis doctoral

7.2.1 Eficiencia de las respuestas usando Redes Neuronales

Una de las contribuciones de la tesis se ha basado en el uso de técnicas de Aprendizaje Automático con el fin de entrenar un AIRS. Concretamente el desarrollo se basa en el Algoritmo

de Retropropagación para medir la eficiencia de cada una de las respuestas ejecutadas por el sistema y retroalimentar el AIRS con el objetivo de que dinámicamente sea capaz de adaptar la selección de respuestas frente a los ataques.

El uso del aprendizaje supervisado hace que vaya guiado por muestras previas cogidas de la experiencia y así el algoritmo converge de manera más rápida. Con este método es posible clasificar cualquier tipo de respuesta, independientemente del ataque en progreso, es decir, que es capaz de medir la eficiencia de la respuesta incluso para nuevos ataques debido a que la entrada de la Red Neuronal es el grado de anomalía del contexto. Además el resultado obtenido de la Red Neuronal es independiente del activo afectado por el ataque siendo así fácil la integración de este sistema en cualquier organización.

7.2.2 Sistema de predicción basado en HMMs

Durante este trabajo de tesis se ha demostrado la viabilidad del sistema de predicción propuesto basado en HMMs en entornos de tiempo real. Este sistema proporciona detección temprana de ataques previendo los pasos de los atacantes. En concreto se propone el uso de un modelo HMM para modelar procesos estocásticos no observables de alertas reportadas desde IDSs para cada tipo de ataque multi-paso.

Para el desarrollo de esta contribución de tesis se ha llevado a cabo los siguientes pasos:

1. Estudio de distintos modelos matemáticos usados para la predicción de ataques multi-paso (2.6)
2. Como contribución de tesis se define un modelo HMM extendido $\lambda_k = (\Sigma, S, A, Q, \pi, H)$ para cada ataque multi-paso.
    - Los estados se corresponden con los pasos similares de un tipo de ataque multi-paso.
    - Otra de las contribuciones de la tesis es el método de obtención de las observaciones del modelo, Σ. Este método se basa en diferentes etiquetas obtenidas a partir del repositorio de CVE y de la severidad de la alerta. De este modo, se evitan po-

sibles sobreajustes del modelo y hace posible que el sistema sea capaz de predecir ataques desconocidos para los modelos HMM que componen dicho sistema.

- El modelo se debe entrenar previamente con el mayor conjunto de datos posibles de cada uno de los estados de todos los posibles ataques, de tal manera que aunque un atacante intente usar un ataque modificado conduciría a la misma ruta, incluso si el atacante conoce nuestro modelo.

Para validar el sistema propuesto en la tesis doctoral se ha diseñado el modelo HMM extendido usando escenarios de ataque de DDoS obteniendo como resultado un *HMM configuration file* para dicho tipo de ataque multi-paso. Para este caso de estudio los parámetros del modelo HMM extendido (λ_{ddos}) son:

1. El conjunto Σ está compuesto por las observaciones obtenidas del proceso de clusterización propuesto como contribución de la tesis doctoral, véase sección 6.6.1.1

2. Los estados de la cadena de Markov se muestran en la Figura 6.6.3 y se corresponden con *intrusion attempt*, *Buffer Overflow*, y *denial of service*. Hay que tener en cuenta que esta cadena ha sido simplificada a ataques de *Buffer Overflow* para ganar acceso a sistemas remotos.

3. Las matrices y vectores de probabilidad del modelo han sido entrenados utilizando *DARPA 2000 datasets* en modo supervisado y no supervisado, véase en la sección 6.6.1.3.

La evaluación del modelo λ_{ddos} se ha llevado a cabo mediante el estudio de diferentes factores necesarios para la validación del sistema propuesto:

1. Algoritmo de entrenamiento: La Figura 6.6.5 y la Figura 6.6.6 muestran los resultados aplicando entrenamiento supervisado y no supervisado respectivamente. Ambas gráficas prueban que el modelo entrenado en modo supervisado es capaz de detectar todas las fases del ataque de DDoS, pero es necesario utilizar una mayor cantidad de escenarios en el conjunto de entrenamiento para la aplicación del algoritmo no supervisado de Baum Welch.

2. Predicción de ataques multi-paso: En primer lugar es necesario calcular la probabilidad de encontrarse en cada uno de los estados de la cadena de Markov. Utilizando la mayor probabilidad obtenida en el paso previo y el número de observaciones es posible aplicar la fórmula (6.2) para predecir el avance del ataque multi-paso. En la sección 6.6.3 se analiza la probabilidad obtenida en cada una de las fases de ataque, comprobando que todos los pasos del ataque de DDoS son detectados con alta probabilidad. Finalmente, la sección 6.6.3 demuestra la capacidad de predicción del sistema propuesto, teniendo en cuenta dos posibles escenarios:

    - Ataques conocidos por el modelo: La Figura 6.6.9 muestra los estados y la probabilidad de que se produzca finalmente el ataque de DDoS utilizando las trazas de uno de los escenarios utilizados durante el proceso de entrenamiento.
    - Ataques desconocidos para el modelo: Usando un escenario de ataque de DDoS no aplicado durante el proceso de entrenamiento se valida el sistema propuesto con ataques desconocidos o nuevos para el sistema de predicción. La Figura 6.6.10 muestra los resultados obtenidos con las trazas de ataque obtenidos a partir del escenario de ataque virtual descrito en la sección 6.6.2. Estos resultados demuestran que la solución propuesta es capaz de predecir el ataque multi-paso y por tanto se valida la adaptabilidad del sistema frente a nuevas alertas y vulnerabilidades de los sistemas sin necesidad de realizar un nuevo entrenamiento del modelo.

A continuación se analizan las características aportadas por el sistema de predicción propuesto:

- La probabilidad de ataque obtenida en tiempo real por el sistema predictivo incrementa a medida que el atacante avanza hacia su objetivo final, de esta forma es posible establecer políticas de respuesta a medida de cada organización, como por ejemplo, realizar una respuesta con mayor o menor grado de impacto en función del activo afectado por el ataque multi-paso.

- La implementación llevada a cabo considera que no siempre los IDSs reportan todas las alertas que suceden en el momento del ataque. En concreto, si los IDSs no reportan alertas para alguna de las fases de un ataque multi-paso, ya sea debido a vulnerabilidades de *o-day* o por una mala configuración en los propios IDSs, se considera que se ha producido un *falso negativo* en un estado específico de la cadena de Markov. Uno de los beneficios de la propuesta de tesis doctoral es que todos los pasos del ataque son independientes en el modelo de Markov, de manera que un *falso negativo* no afecta al cálculo de la probabilidad de los siguientes estados de la cadena y por tanto el modelo HMM encajará con el escenario de ataque multi-paso en progreso pudiendo predecir el ataque.

- El modelo es capaz de predecir ataques no incluidos en los escenarios del conjunto de entrenamiento, siendo fácilmente adaptable a nuevas vulnerabilidades incorporadas en los reportes de CVE sin necesidad de realizar un nuevo proceso de entrenamiento offline en el sistema para la detección de nuevas intrusiones.

- Los resultados obtenidos en esta contribución de tesis frente a los trabajos previos en esta línea de investigación estudiados en la sección 2.6.3 proporcionan una mayor precisión en la información en tiempo real sobre el estado en que se encuentra el ataque en progreso. Esta información es vital para la toma de decisiones frente al ataque en forma de respuesta proactiva. Por ejemplo, en [KEAA14] calculan la probabilidad de riesgo del sistema, sin dar mayor información sobre el estado en que se encuentra el ataque o el progreso del atacante hacia su objetivo.

- La definición de todos los parámetros del modelo se basan en un aprendizaje previo, en cambio la predicción llevada a cabo en el trabajo de investigación [SDJC12] depende directamente del parámetro K el cual es asignado manualmente para la optimización de alertas.

- Otra ventaja aportada por el sistema predictivo propuesto frente a otras contribuciones previas estudiadas en la sección 2.6 es la posibilidad de despliegue del sistema en

cualquier organización independientemente de la topología de red.

- El modelo propuesto puede contribuir en distintos ámbitos de seguridad:
  - Modificación del valor de riesgo en Sistemas de Evaluación de Riesgo Dinámico a partir del valor de probabilidad obtenido en tiempo real del ataque multi-paso, incrementando el riesgo a medida que el atacante avanza hacia su objetivo final.
  - Ejecución de respuestas proactivas frente a ataques multi-paso en tiempo real en un AIRS teniendo en cuenta el valor de probabilidad de dicho ataque.
  - Análisis forense tras un ataque para la detección de fases a partir de las trazas de dicho ataque.
  - Descubrimiento de nuevos escenarios de ataque mediante el análisis a posteriori de las probabilidades obtenidas en cada etapa detectada por el modelo.

7.3 Líneas futuras de investigación

7.3.1 Eficiencia de las respuestas usando Redes Neuronales

En el trabajo desarrollado durante la tesis doctoral ha quedado pendiente el estudio de las distintas topologías de Red Neuronal para validar y seleccionar la topología mejor para el cálculo de la efectividad de la respuesta utilizando la metodología de validación cruzada.

Validar los dos tipos de funciones de de activación sigmoidal incluidas en la implementación (véase 5.2.2.1) y los dos métodos implementados para evitar la convergencia del algoritmo en mínimos locales (Factor de aprendizaje adaptativo y Aprendizaje con momentos).

Por otro lado habría que analizar el tiempo de estabilización (Figura 5.2.1) de cada uno de los parámetros del contexto para que el módulo *Context Receiver* pueda tomar las mediciones en el momento en el que la respuesta lanzada por el AIRS tenga el efecto necesario en el sistema para que vuelva al *contexto normal*.

Otras mejoras que se pueden realizar sobre el algoritmo de Retropropagación son:

- La inicialización de pesos basada en rango.

- Utilizar el método SAB para evitar los mínimos locales de la función de error. Este método se basa en la combinación de los métodos de Factor de aprendizaje adaptativo y Aprendizaje con momentos.

Por otro lado, se pueden analizar otros algoritmos de aprendizaje automático, como por ejemplo el algoritmo bio-inspirado basado en el sistema inmune.

7.3.2 Sistema de predicción basado en HMMs

Una de las primeras líneas futuras es la validación del sistema propuesto con un mayor número de escenarios de prueba de DDoS. Posteriormente, modelar distintos HMMs con otros tipos de ataques multi-paso para completar la validación del sistema frente a distintos ataques.

El trabajo llevado a cabo durante la tesis doctoral deja abiertas distintas vías de continuación:

- Análisis de posibles respuestas proactivas considerando políticas de la organización y la importancia de los activos que componen la red corporativa.
- Las transiciones y la probabilidad de ataque calculada por del modelo HMM se basan en las observaciones obtenidas de las alertas del ataque multi-paso. Como trabajo futuro se podría integrar la efectividad de la respuesta para determinar si se producen nuevas transiciones entre estados de la cadena de Markov o modificaciones en la probabilidad de ataque. Para ello es necesario añadir nuevas observaciones a los modelos HMM que representen dichas respuestas. De esta forma es posible representar las interacciones entre el atacante y el sistema de respuesta.
- Un ataque podría producir un número de alertas diferentes dependiendo de la experiencia del atacante. Como trabajo futuro se puede contemplar la posibilidad de tener varios vectores de número medio de observaciones de acuerdo a la experiencia del atacante.

- Determinar cuando las alertas son producidas por un nuevo ataque o por el ataque en progreso utilizando métodos de agregación y filtrado de alertas.
- Integrar y validar las contribuciones de la tesis en los distintos ámbitos de seguridad indicados en la sección 7.2

7.4 Difusión de resultados

Este trabajo de tesis doctoral ha sido posible gracias al conocimiento adquirido a partir de varios proyectos de investigación listados en la Tabla 7.4.1.

A partir de la investigación llevada a partir de estos proyectos se ha obtenido como resultado las siguientes publicaciones:

- Publicaciones sobre el cálculo de la eficiencia de respuestas (contribución 1):
  - Neural Networks applied to the learning process of Automated Intrusion Response Systems, JNIC 2015
  - Redes Neuronales aplicadas al proceso de aprendizaje de un Sistema de Respuesta a Intrusiones Automático, Jitel 2013
- Publicaciones sobre la predicción de ataques multi-paso (contribución 2):
  - Real-time multistep attack prediction based on Hidden Markov Models - IEEE Transactions on Dependable and Secure Computing (Revista JCR - Q1) -14 páginas - 2017.
  - Sistema de detección de fases de ataque basado en Modelos Ocultos de Markov, JNIC 2016
- Publicaciones relacionadas con ambas contribuciones de la tesis doctoral:
  - Evolving from a static toward a proactive and dynamic risk-based defense strategy, JNIC 2015

Proyecto	Financiación	Fechas	Actividad
SEGUR@ (Security and Trust in the Information Society)	CDTI, Ministerio Español de Ciencia e innovación bajo el programa CENIT	Julio 2007 hasta Diciembre 2010	Desarrollo de una arquitectura colaborativa de seguridad entre SIMS (Sistemas de gestión de información de seguridad) mediante Ontologías. Uso de librerías de Ontologías en JAVA y MySQL. (Ontología IDMEF, Jena, SPARQL, Joseki, RDF. Inferencias con Jena2, Pellet y Protégé mediante reglas SWRL). Mecanismos de análisis de seguridad contra Botnets y SPAM.
RECLAMO (Red de Sistemas de Engaño Virtuales y Colaborativos basados en Sistemas Autónomos de Respuesta a Intrusiones y Modelos de Confianza)	MICINN Español (códigos TIN2011-28287-C02-01 y TIN2011-28287-C02-02) y la Comisión Europea (FEDER/ERDF)	Enero 2012 hasta Diciembre 2014	Desarrollo de un algoritmo de aprendizaje para la automatización y mejora de respuestas a intrusiones usando una Red Neuronal Artificial e implementación de un parser de alertas IDMEF en JAVA.
DHARMA (Dynamic Heterogeneous threAts Risk Management and Assessment)	MINECO Español (código TIN2014-59023-C2-2-R) y la Comisión Europea (FEDER/ERDF)	Enero 2015 hasta Diciembre 2017	Desarrollo de un sistema de predicción de intrusiones basado en HMM (Hidden Markov Model) y vulnerabilidades (CVE, Common Vulnerabilities Exposures) asociadas a las alertas reportadas por los IDSs (Intrusion Detection System).
VISU (Herramienta de visualización y trazado de un ciberataque)	Convenio MCCD-UPM	Enero 2016 hasta Diciembre 2016	Integración del modelo predictivo basado en HMM con el resto del sistema desarrollado en el proyecto para la gestión de alertas.
DRONE-FS (Self-protected Filesystem for Drones and devices including confidential information)	MINECO (código RTC-2015-4064-8)	Septiembre 2015 hasta Diciembre 2017	Gestión técnica y diseño del proyecto. Diseño del módulo de cifrado para la protección de ficheros en un entorno DLP y la supervisión de la implementación de retos multimedia en entornos móviles Android.
CYBERNOID (Cybersecurity in Critical Infrastructures with Android devices)	MINETUR (código TSI-100200-2015-035)	Septiembre 2015 hasta Diciembre 2017	Gestión técnica y diseño del proyecto. Diseño del servidor externo (http-rest) para la autenticación y ejecución de diversos retos, con el objetivo de retornar un JSON con una serie de sub-claves que componen la clave de cifrado simétrico de cada uno de los ficheros para proporcionar protección contra fugas de información en entornos DLP.
MODRIC (Modelo de Gestión de Riesgos Global e Infraestructuras Críticas)	Convenio INCIBE-UPM	Noviembre 2016 hasta Octubre 2018	Gestión técnica, temporal y diseño del proyecto. Análisis de datos en formato CEF. Realización del estado del arte sobre gestión y análisis del riesgo dinámico.

Tabla 7.4.1: Proyectos de investigación

  - Sistema de Respuesta a Intrusiones Proactivo basado en Modelos de Markov y Redes Neuronales, CyberCamp 2014

- Otras publicaciones:
  - Sistema de cifrado basado en contexto aplicado a prevención de fuga de datos, Jitel 2017
  - Context-based encryption applied to Data Leakage Prevention solutions, SECRYPT 2017 (CORE B)
  - Sistemas de estimación de Botnets en RedIRIS, Jornadas RedIRIS 2010
  - Sharing information about security alerts using semantic web technologies, CNSM 2010 (CORE B)
  - A semantic web approach to share alerts among Security Information Management Systems, IBWAS 2009

Bibliografía

[A+80] James P Anderson et al. Computer security threat monitoring and surveillance. Technical report, Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, 1980.

[Amo99] Edward Amoroso. Intrusion detection: an introduction to internet surveillance, correlation, trace back, traps, and response. *Intrusion. Net Book*, 1999.

[APFC10] Nor Badrul Anuar, Maria Papadaki, Steve Furnell, and Nathan Clarke. An investigation and survey of response options for intrusion response systems (irss). In *Information Security for South Africa (ISSA), 2010*, pages 1–8. IEEE, 2010.

[AZC09] M Ali Aydın, A Halim Zaim, and K Gökhan Ceylan. A hybrid intrusion detection system design for computer network security. *Computers & Electrical Engineering*, 35(3):517–526, 2009.

[CC16] CCN-CERT. CCN-CERT IA-09/16 Ciberamenazas 2015 / Tendencias 2016. [Online] https://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos/1483-ccn-cert-ia-0916-ciberamenazas-2015-tendencias-2016-resumen-ejecutivo/file.html, 2016.

[Cer14] Certsi. Ataques de denegación de servicio vía NTP. [Online] https://www.certsi.es/blog/ataques-dos-ntp, 2014.

[CM02] Frédéric Cuppens and Alexandre Miege. Alert correlation in a cooperative intrusion detection framework. In *ieee symposium on Security and privacy.*, pages 202–215. IEEE, 2002.

[Col12] Eric Cole. *Advanced persistent threat: understanding the danger and how to protect your organization*. Newnes, 2012.

[Col13] R Cole. Multi-step Attack Detection via Bayesian Modeling Under Model Parameter Uncertainty. 2013.

[Com16] Grisworld & Company. Artificial Neural Networks and Deep Learning – The True Key to Artificial Intelligence. [Online] http://griswoldandco.com/artificial-neural-networks-and-deep-learning-the-true-key-to-artificial-intelligence/, 2016.

[Cor17] Symantec Corp. Internet security threat report (istr), 2017.

[cve] MITRE Inc. [Online] http://www.cve.mitre.org/.

[DAR00] DARPA. DARPA - intrusion detection evaluation dataset. [Online] https://www.ll.mit.edu/ideval/data/2000data.html, 2000.

[DCF] H Debar, D Curry, and B Feinstein. The intrusion detection message exchange format (IDMEF). ietf rfc 4765.

[Des03] Neil Desai. Intrusion prevention systems: The next step in the evolution of ids. *Security Focus*, 27, 2003.

[DHA15] Proyecto Dharma. [Online] http://dharma.inf.um.es/, 2015.

[DLHY10] Haitao Du, Daniel F Liu, Jared Holsopple, and Shanchieh Jay Yang. Toward Ensemble Characterization and Projection of Multistage Cyber Attacks. In *ICCCN*, pages 1–8, 2010.

[DLR77] A P Dempster, N M Laird, and D B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[Dub02] Denise Dubie. Users shoring up net security with sim. *Network World*, 19(39):21–21, 2002.

[DVGVB09] Jorge E López De Vergara, Antonio Guerrero, Víctor A Villagrá, and Julio Berrocal. Ontology-based network management: study cases and lessons learned. *Journal of Network and Systems Management*, 17(3):234–254, 2009.

[dVVM+09] Jorge E López de Vergara, Enrique Vázquez, Antony Martin, Samuel Dubus, and Marie-Noëlle Lepareux. Use of ontologies for the definition of alerts and policies in a network security platform. *JNW*, 4(8):720–733, 2009.

[DW01] Hervé Debar and Andreas Wespi. Aggregation and correlation of intrusion-detection alerts. In *International Workshop on Recent Advances in Intrusion Detection*, pages 85–103. Springer, 2001.

[Dyn16] Dyn. Dyn Analysis Summary. [Online] http://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack, 2016.

[FBY08] Daniel S Fava, Stephen R Byers, and Shanchieh Jay Yang. Projecting cyberattacks through variable-length markov models. *IEEE Transactions on Information Forensics and Security*, 3(3):359–369, 2008.

[For73] G D Forney. The Viterbi algorithm. In *Proceeding IEEE*, 1973.

[FWM+05] Bingrui Foo, Y-S Wu, Y-C Mao, Saurabh Bagchi, and Eugene Spafford. Adepts: adaptive intrusion response using attack graphs in an e-commerce environment. In *International Conference on Dependable Systems and Networks, DSN.*, pages 508–517. IEEE, 2005.

[FWZZ09] L Feng, W Wang, L Zhu, and Y Zhang. Predicting intrusion goal using dynamic Bayesian network with transfer probability estimation. *Journal of Network and Computer*, 2009.

[GFDVC10] Fermín Galán, David Fernández, Jorge E López De Vergara, and Ramon Casellas. Using a model-driven architecture for technology-independent scenario configuration in networking testbeds. *IEEE Communications Magazine*, 48(12):132–141, 2010.

[HA03] X Hu and ES Atwell. A survey of machine learning approaches to analysis of large corpora. In *Proceedings of SProLaC: Workshop on Shallow Processing of Large Corpora*, pages 45–52. UCREL, Lancaster University, 2003.

[HAK07] K Haslum, A Abraham, and S Knapskog. DIPS: A Framework for Distributed Intrusion Prediction and Prevention Using Hidden Markov Models and Online Fuzzy Risk. In *Third International Symposium on Information Assurance and Security (IAS). IEEE*, 2007.

[Hea08] Jeff Heaton. *Introduction to neural networks with Java*. Heaton Research, Inc., 2008.

[Her05] Cristhian Herrera. Fundamentos básicos del reconocimiento de Voz. [Online] https://www.adictosaltrabajo.com/tutoriales/complementosr/, 2005.

[HMK08] K Haslum, M E Moe, and S Knapskog. Real-time intrusion prevention and security analysis of networks using HMMs. In *33rd IEEE Conference on Local Computer Networks (LCN).*, 2008.

[HPSB+04] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21:79, 2004.

[HVV17] Pilar Holgado, Víctor A. Villagrá, and Luis Vazquez. Real-time multistep attack prediction based on hidden markov models. *IEEE Transactions on Dependable and Secure Computing*, 2017.

[IS12] Aderonke Justina Ikuomola and Adesina Simon Sodiya. A credible cost-sensitive model for intrusion response selection. In *Fourth International Conference on Computational Aspects of Social Networks (CASoN)*, pages 222–227. IEEE, 2012.

[jah] Jahmm - Java library for HMM model and algorithms. [Online] https://code.google.com/p/jahmm/.

[KEAA14] H A Kholidy, A Erradi, S Abdelwahed, and A Azab. A finite state Hidden Markov Model for predicting multistage attacks in Cloud Systems. In *12th International Conference on Dependable, Autonomic and Secure Computing (DASC). IEEE*, pages 14–19, 2014.

[Ken99] K Kendall. *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems*. PhD thesis, 1999.

[KO11] Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011.

[Koh88] Teuvo Kohonen. An introduction to neural computing. *Neural networks*, 1(1):3–16, 1988.

[KZD+15] V Katkar, A Zinjade, S Dalvi, T Bafna, and R Mahajan. Detection of DoS/DDoS Attacks against HTTP Servers using Naive Bayesian. In *International Conference on Computing Communication Control and Automation (ICCUBEA). IEEE*, pages 280–285, 2015.

[LCV03] U Lindqvist, S Cheung, and R Valdez. Correlated attack modeling (cam). 2003.

[Mac10] N Macari. Analysis of a machine learning algorithm and corpus as a tool for managing the ambiguity problem of search engines. *Master of Science, Fakultat Informatik, Technische Universitat Dresden*, 2010.

[MAHZ12] Zeenat Mahmood, Chetan Agrawal, Syed Shadab Hasan, and Syeda Zenab. Intrusion detection in cloud computing environment using neural network. *International Journal of Research in Computer Engineering & Electronics*, 1(1):19–22, 2012.

[Mic17] Microsoft. WannaCry. [Online] https://blogs.technet.microsoft.com/mmpc/2017/05/12/wannacrypt-ransomware-worm-targets-out-of-date-systems/, 2017.

[ML13] Verónica Mateos Lanchas. Contribución a la automatización de sistemas de respuesta frente a intrusiones mediante ontologías. 2013.

[MLVGB10] Verónica Mateos Lanchas, Víctor Villagrá González, and Francisco Bueno. Ontologies-based automated intrusion response system. *Computational Intelligence in Security for Information Systems 2010*, pages 99–106, 2010.

[MM14] Lino Francisco Manjarrez Montaño. Relaciones neuronales para determinar la atenuación del valor de la aceleración máxima en superficie de sitios en roca para zonas de subducción. 2014.

[MM17] Arnulfo Mateo Mateo. Cerebro y Neurociencia. [Online] http://cerebroyneurociencia.blogspot.com.es/2017/07/de-la-neurona-al-cerebro.html, 2017.

[MVH+04] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

[MVRB12] Verónica Mateos, Víctor A Villagrá, Francisco Romero, and Julio Berrocal. Definition of response metrics for an ontology-based automated intrusion response systems. *Computers & Electrical Engineering*, 38(5):1102–1114, 2012.

[NAAS12] Reyadh Shaker Naoum, Namh Abdula Abid, and Zainab Namh Al-Sultani. An enhanced resilient backpropagation artificial neural network for intrusion detection system. *International Journal of Computer Science and Network Security (IJCSNS)*, 12(3):11, 2012.

[NCR02] Peng Ning, Yun Cui, and Douglas S Reeves. Analyzing intensive intrusion alerts via correlation. In *International Workshop on Recent Advances in Intrusion Detection*, pages 74–94. Springer, 2002.

[PXG+12] Ling-xi Peng, Dong-qing Xie, Ying Gao, Wen-bin Chen, Fu-fang Li, and Wu Wen. An inmune-inspired adaptative automated intrusion response system. international journal of computational intelligence. *Systems*, 5:5, 2012.

[qua] Qualys, Inc. [Online] https://blog.qualys.com/laws-of-vulnerabilities/2015/03/17/ghost-remote-code-execution-exploit.

[Rab85] J Rabiner. A Probabilistic Distance Measure for HMMs. *AT&T Technical Journal*, 64:391–408, 1985.

[Rab89] L R Rabiner. A tutorial of Hidden Markov Models and selected applications in speech recognition. In *Procedings of the IEEE*, 1989.

[RHW85] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[ROC05] Michael Rash, Angela Orebaugh, and Graham Clark. *Intrusion prevention and active response: deploying network and host IPS*. Syngress, 2005.

[SBW07] Natalia Stakhanova, Samik Basu, and Johnny Wong. A taxonomy of intrusion response systems. *International Journal of Information and Computer Security*, 1(1-2):169–184, 2007.

[SDJC12] A S Sendi, M Dagenais, M Jabbarifar, and M Couture. Real time intrusion prediction based on Optimized Alerts with Hidden Markov Model. *Journal of Networks*, 2012.

[Sen13] A Shameli Sendi. System health monitoring and proactive response activation. 2013.

[Sha01] Claude E Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[SM13] F Seraj and C Meinel. Attack scenario prediction methodology. In *Tenth International Conference on Information Technology: New Generations (ITNG). IEEE*, 2013.

[sno] Snort - Network intrusion detection system. [Online] https://www.snort.org/.

[SSD14] A Shameli-Sendi and M Dagenais. ARITO: Cyber-attack response system using accurate risk impact tolerance. *International journal of information*, 2014.

[SSD15] A Shameli-Sendi and M Dagenais. ORCEF: online response cost evaluation framework for intrusion response system. *Journal of Network and Computer*, 2015.

[SSDDJ13] A Shameli-Sendi, J Desfossez, M Dagenais, and M Jabbarifar. A Retroactive-Burst Framework for Automated Intrusion Response System. *Journal of Computer Networks and communications*, 2013.

[SSEJJD12] Alireza Shameli-Sendi, Naser Ezzati-Jivan, Masoume Jabbarifar, and Michel Dagenais. Intrusion response systems: survey and taxonomy. *IJCSNS*, 12(1):1–14, 2012.

[Sta04] William Stallings. *Fundamentos de seguridad en redes: aplicaciones y estándares*. Pearson Educación, 2004.

[VGML09] Víctor A Villagrá González and Verónica Mateos Lanchas. *Seguridad en redes de telecomunicación*. Fundación Rogelio Segovia para el Desarrollo de las Telecomunicaciones, 2009.

[vnx] VNX. [Online] http://dit.upm.es/vnxwiki/index.php.

[WFM+07] Yu-Sung Wu, Bingrui Foo, Yu-Chun Mao, Saurabh Bagchi, and Eugene H Spafford. Automated adaptive intrusion containment in systems of interacting services. *Computer networks*, 51(5):1334–1360, 2007.

[Wik17] Wikipedia. Modelo Oculto de Markov. [Online] https://es.wikipedia.org/wiki/Modelo_oculto_de_Márkov, 2017.

[YF07] D Yu and D Frincke. Improving the quality of alerts and predicting intruder's next goal with Hidden colored Preti-Net. *Computer Networks*, 2007.

[YK03] Shun-Zheng Yu and Hisashi Kobayashi. An efficient forward-backward algorithm for an explicit-duration hidden Markov model. *Signal Processing Letters, IEEE*, 10(1):11–14, 2003.

[ZGHS09] Xin Zan, Feng Gao, Jiuqiang Han, and Yu Sun. A Hidden Markov Model based framework for tracking and predicting of attack intention. In *International Conference on Multimedia Information Networking and Security. MINES'09.*, volume 2, pages 498–501. IEEE, 2009.

[ZKSY14] S A Zonouz, H Khurana, W H Sanders, and Yardley. RRE: A game-theoretic intrusion response and recovery engine. In *IEEE Transactions on Parallel and Distributed Systems*, volume 25, pages 395–406, 2014.

[ZRB+12] Saman Zonouz, Katherine M Rogers, Robin Berthier, Rakesh B Bobba, William H Sanders, and Thomas J Overbye. SCPSE: Security-oriented cyber-physical state estimation for power grid critical infrastructures. *IEEE Transactions on Smart Grid*, 3(4):1790–1799, 2012.