Sistema para la correlación de alertas de NIDS basados en anomalías

Tesis para el Máster de Investigación

Jorge Maestre Vidal

Director

Luis Javier García Villalba

Calificación: SOBRESALIENTE

Facultad de Informática

Universidad Complutense de Madrid

Madrid, Junio de 2013

Agradecimientos

Los autores agradecen la financiación que les brinda el Subprograma AVANZA COMPETITIVIDAD I+D+I del Ministerio de Industria, Turismo y Comercio (MITyC) a través del Proyecto TSI-020100-2011-165. Asimismo, los autores agradecen la financiación que les brinda el Programa de Cooperación Interuniversitaria de la Agencia Española de Cooperación Internacional para el Desarrollo (AECID), Programa PCI-AECID, a través de la Acción Integrada MAEC-AECID MEDITERRÁNEO A1/037528/11.

Abstract

The deployment of a Network-based Intrusion Detection System (NIDS) in the system perimeter security should be supplemented with tools to help improve their behavior. The alert correlation systems are designed to facilitate the management of alerts issued, providing additional information and allowing their aggregation, thereby expediting prevention processes

Throughout the years, the research area of the alert correlation focused on distributed attack detection. The main motivation of this work derived from the lack of proposals for correlating the alerts from anomaly-based NIDS, which process the audited traffic payload. To do this, the proposed system performs a labeling of the alerts issued by the NIDS, taking into account both quantitative and qualitative criteria. Also, is considered the analysis performed at the packet-level, and the analysis performed at the trace-level, as an alternative to the construction of attack scenarios.

In view of the results obtained from the experiments, the objectives have been met satisfactorily. The proposals have managed to mitigate the problem of false-positive injection and has managed to group the NIDS alerts based on the established criteria. In addition, the impact on the protected system has been minimal, ensuring traffic processing in real time.

Keywords

Anomalies, Alert Correlation, Intrusion, NIDS.

Resumen

El despliegue de un Sistema de Detección de Intrusiones basado en Redes (NIDS) en el perímetro de seguridad de un sistema ha de complementarse con la incorporación de herramientas que ayuden a mejorar su comportamiento. Los sistemas de correlación de alertas tienen como objetivo facilitar la gestión de las alertas emitidas, aportando información adicional, y permitiendo su agregación, agilizando de esta manera los procesos de prevención.

A lo largo de los años, el área de investigación en la correlación de alertas se ha centrado en la detección de ataques distribuidos. La principal motivación de este trabajo procede de la escasez de propuestas para correlacionar las alertas emitidas por NIDS basados en anomalías, que establecen como objeto del análisis, la carga útil del tráfico auditado. Para ello, el sistema propuesto lleva a cabo un etiquetado de las alertas emitidas por el NIDS, tomando en cuenta criterios cuantitativos y cualitativos. Además, se considera el análisis llevado a cabo a nivel de paquetes, y el análisis llevado a cabo a nivel de trazas como alternativa a la reconstrucción de escenarios de ataques.

A la vista de los resultados arrojados por los experimentos, los objetivos han sido cumplidos satisfactoriamente. Las propuestas han conseguido mitigar el problema de la inyección de falsos positivos y se ha conseguido agrupar las alertas emitidas, en base a los criterios establecidos. Además, el impacto sobre el sistema protegido ha sido mínimo, garantizando un procesamiento del tráfico en tiempo real.

Palabras clave

Anomalías, Correlación de alertas, Intrusión, NIDS

Autorizo a la Universidad Complutense de Madrid difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Jorge Maestre Vidal

Lista de Acrónimos

ACC	*Aggregation and Correlation Component*
ACM	*Alert Correlation Matrix*
APACS	*Anomaly Payload Alert Correlation System*
API	*Application Programming Interface*
BIDS	*Battlefield Intrusion Detection System*
CIDF	*Common Intrusion Detection Framework*
CISL	*Common Intrusion Specification Language*
DARPA	*Defense Advanced Research Projects Agency*
DNS	*Domain Name System*
FPGA	*Field Programmable Gate Array*
FSM	*Frequent Structure Mining*
GIDO	*General Intrusion Detection Object*
HIDS	*Host-based Intrusion Detection System*
ICMP	*Internet Control Message Protocol*
IDMEF	*Intrusion Detection Message Exchange Format*
IDS	*Intrusion Detection System*
IDWG	*Intrusion Detection Working Group*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
IPS	*Intrusion Prevention System*
LAMBDA	*Lenguage to Model a Databases for Detection of Attacks*
LIDS	*Linux Intrusion Detection System*
MAC	*Mandatory Access Control*
MLP	*Multilayer Perceptron*
MTU	*Maximum Transmission Unit*
NIDS	*Network-based Intrusion Detection System*
NTP	*Network Time Protocol*
R2R	*Remote To Root*
ROC	*Receiver Operating Characteristic*
SO	*Sistema Operativo*
SOM	*Self-Organizing Map*
SVM	*Support Vector Machines*
TEC	*Tivoli Enterprise Console*
TCP	*Transmission Control Protocol*
U2R	*User To Root*
UCM	*Universidad Complutense de Madrid*
UDP	*User Datagram Protocol*
VoIP	*Voice over Internet Protocol*
XML	*Extensible Markup Language*

Índice General

Índice de Tablas

Índice de Figuras

Capítulo 1

Introducción

Un Sistema de Detección de Intrusiones o IDS (*Intrusion Detection System*) es un mecanismo de defensa que basa su comportamiento en el análisis de los diferentes eventos que acontecen en el sistema protegido buscando indicios de actividades maliciosas. El IDS clasifica los eventos en legítimos e ilícitos, siendo estos últimos considerados como intrusiones. Si un IDS además de detectar tiene la capacidad de llevar a cabo acciones para evitar o mitigar los efectos de la intrusión, se denomina Sistema de Prevención de Intrusos o IPS (*Intrusion Prevention System*). Cuando el IDS opera en un entorno de red se dice que es un Sistema de Detección de Intrusos en Redes conocido como NIDS (*Network-based Intrusion Detection System*) y si lo hace a nivel local (*host*), es un Sistema de Detección de Intrusos en *Host* conocido como HIDS (*Host-based Intrusion Detection System*). Cualquiera de ellos puede adoptar el comportamiento de prevención, además del de detección.

Pero el despliegue de un IDS plantea nuevos inconvenientes: por un lado, el procesamiento de información puede llegar a penalizar la capacidad de reacción del sistema en tiempo real; Además, cuando el sistema protegido se expone a entornos potencialmente hostiles, el IDS tiene tendencia emitir una gran cantidad de alertas. Esta última situación dificulta la capacidad del operador de tomar las medidas de prevención adecuadas. Para simplificar el tratamiento de las alertas, el perímetro de defensa incorpora sistemas de correlación de alertas. El objetivo de dichos sistemas es clasificar y agrupar las alertas emitidas por uno o varios IDS, con el fin de facilitar su tratamiento y prevención.

El resto de este capítulo está organizado como sigue: El apartado 1.1 presenta el objeto de investigación. En el apartado 1.2 se analizan trabajos relacionados. En último lugar, el apartado 1.3 resume la estructura del resto del trabajo

1.1. Objeto de la investigación

La instalación de un IDS en el perímetro de seguridad de un sistema ha de complementarse con la incorporación de herramientas que ayuden a mejorar su comportamiento. Los sistemas de correlación de alertas tienen como objetivo aumentar el conocimiento obtenido de un intento de intrusión, a partir de la información que proveen las alertas generadas por el IDS. Este conocimiento va a permitir una clasificación del tráfico en base a diferentes criterios, ayudando a los operadores a tomar las mejores decisiones y mejorando aquellas que se llevan a cabo en tiempo real.

La motivación del proyecto ha sido llevar a cabo un diseño capaz de correlacionar las alertas emitidas por NIDS basados en anomalías que analizan la carga útil del tráfico de la red, en busca *malware*. Los principales objetivos han sido la identificación de los falsos positivos, la valoración cualitativa de la probabilidad de que una anomalía sea efectivamente una amenaza y una clasificación cuantitativa que muestre el tipo concreto de amenaza detectada. Esta última clasificación es llevada a cabo considerando ataques a nivel de paquete o ataques a nivel de trazas de tráfico.

Además, el sistema propuesto ha de ser capaz de trabajar en tiempo real, produciendo un impacto mínimo sobre el sistema protegido. Como objetivo adicional, se propone llevar a cabo un despliegue sobre un escenario real de pruebas.

1.2. Trabajos relacionados

En la especificación del estándar para el intercambio de alertas entre IDS conocido como IDMEF (*Intrusion Detection Message Exchange Format*) [1], cuando se habla de correlación de intrusiones se hace distinción entre dos tipo correlaciones: la correlación que se lleva a cabo entre eventos y la correlación que se lleva a cabo entre alertas. También explica que la correlación de eventos hace referencia a eventos neutrales que no necesariamente denotan actividades malintencionadas, mientras que la correlación de alertas hace referencia a amenazas conocidas o a anomalías en el modo de uso habitual y legítimo del sistema protegido. El objetivo de este trabajo se centra en la correlación de alertas y no de eventos, pero resulta interesante resaltar la distinción de uso que propusieron los creadores del estándar.

A lo largo de los años la correlación de alertas ha evolucionado para adaptarse a las nuevas tendencias en el área de intrusiones. Durante los años 2000 y 2004 los trabajos publicados sembraron las bases y las ideas principales para el tratamiento

de las alertas. En los dos años siguientes, la investigación se centró en la reconstrucción de escenarios de ataques. A partir de entonces la correlación de alertas pierde protagonismo cedida por el auge de la investigación en el área de la detección de intrusiones basada en anomalías. Pero a partir del año 2008 vuelven a publicarse interesantes propuestas, centradas en la incorporación de nuevas estrategias de agrupamiento y etiquetado basadas en la minería de datos, y en la necesidad de la correlación de alertas en sistemas distribuidos.

Uno de los primeros trabajos de correlación de alertas, fue llevado a cabo por la empresa IBM. Se trata del sistema ACC (*Aggregation and Correlation Component*) que posteriormente fue presentado en el año 2001 por H. Debar y A. Wespi [2]. Dado que ha sido uno de los trabajos pioneros, su objetivo se basaba en solucionar dos problemas simples, pero importantes: Mitigar la repetición de alertas y organizarlas en función de sus consecuencias aplicando diferentes criterios.

En ese mismo año, se presentó otro de los trabajos más influyentes en el área de correlación de alertas: el trabajo de A. Valdes y K. Skinner [3]. Dicho trabajo supuso la definición de un marco matemático para la correlación de alertas, y fue de los primeros en aplicar el uso de la arquitectura EMERALD para la detección de intrusiones, y de su complementación mediante mecanismos de correlación de alertas. La propuesta sucedía a su trabajo previo [4] en el campo de la detección de intrusiones. Dicha línea de investigación culminaría con el trabajo [5] también aplicado sobre EMERALD, para la correlación de las alertas generadas por diferentes sensores homogéneos del tráfico de una red. Siguiendo la linea de trabajo sobre dicho NIDS, P. Porras et al. [6] extiende su arquitectura al sistema M-Correlator, añadiendo a los criterios de clasificación la importancia del efecto que pueda causar la amenaza sobre sistema víctima.

A la vista de la necesidad del desarrollo de este tipo de sistemas, el MIT también tomó partido. O. M. Dain y R. K. Cunningham [7] presentaron una alternativa a la arquitectura EMERALD, que tenía como principal diferencia la optimización automática de los parámetros de correlación mediante la incorporación de conjuntos de entrenamiento y un nuevo sistema de etiquetado de alertas. Pero dicha propuesta nunca llegó a tener la misma importancia a nivel académico.

A finales del 2002 aparecieron propuestas más originales que derivarían en la gran variedad de aproximaciones que actualmente combaten los problemas de la correla-

ción de alertas. Una de las más innovadoras fue la de Ning and Xu [8] centrada en la optimización del proceso de correlación, mediante el uso de técnicas de indexación avanzadas comúnmente empleadas para mejorar el tiempo de consulta en las bases de datos. Por otro lado, F. Cuppens et al. [9] propusieron modelar la correlación en función del objetivo y el proceso de intrusión. Esta fue la extensión de su trabajo previo [10] para correlacionar los ataques en función a dichos objetivos. Este enfoque resultó interesante a Ning y Xu quienes publicaron en el 2003 un nuevo trabajo [11] que pretendía extraer las estrategias del atacante estableciendo la similitud entre diferentes secuencias de ataques.

Otra rama de investigación inspirada en estos trabajos previos, consiste en la reconstrucción de escenarios de ataques a partir de las alertas generadas por los sensores. En el 2004, Noel et al. publicaron otro importante trabajo [12] con la misma finalidad, basado en la asociación de los grafos que generan los ataques en la red protegida. Dicha estrategia tendría mucho éxito en otra rama muy diferente de la detección de intrusiones: la del análisis y la gestión de riesgos dinámicos. Pero el enfoque que abrieron Ning y Xu para construcción de escenarios de ataques tuvo también muchos seguidores. En el año 2006, Zhu y Ghorbani presentaron otro importante trabajo sobre esa línea [13], que destacó entre las múltiples propuestas que fueron apareciendo.

En el año 2009 Saddodin et al. [14] propusieron un interesante marco para la correlación de alertas en tiempo real, que incorpora nuevas técnicas para la agregación de alertas en patrones estructurales. A partir de estos trabajos comienza a ser muy frecuente el uso de estrategias de *Soft-Computing* para llevar a cabo la agregación de alertas, como por ejemplo el uso de lógica difusa, representada por algunos interesantes trabajos como el de H. Elshoush et al. [15] para la reducción de la tasa de falsos positivos de un IDS, o el de K. Alsubhi et al. [16] para la priorización del tratamiento de alertas.

También han sido frecuentes los trabajos basados en el uso de redes neuronales artificiales, como es el caso del uso de mapas auto-organizativos SOM [17], arboles de decisión y máquinas de vectores de soporte SVM (Support Vector Machines) [18], estrategias de minería de datos y agrupamiento [19], algoritmos genéticos [20] [21] y el uso de agentes [22].

Pero sin duda la tendencia actual en el área de la correlación de alertas son las

propuestas referentes a la correlación de alertas en sistemas de detección distribuidos, orientados a brindar protección a sistemas desplegados mediante arquitecturas de computación *Cloud* y *Ad Hoc* [22] [23] [24]. Esta última tendencia se aleja de la motivación del trabajo realizado.

1.3. Estructura del trabajo

El documento está estructurado en 6 capítulos, con la estructura que se comenta a continuación.

Además de la presente introducción, el segundo capítulo realiza una aproximación a los Sistemas de Detección de Intrusiones, repasando sus características principales, estado del arte y su clasificación. Además se presenta una taxonomía adicional centrada en los NIDS basados en anomalías, realizada en base a las estrategias de modelado del tráfico de la red.

El capítulo 3 aborda el concepto de sistema de correlación de alertas y sus principales características. Se describen las principales etapas en la correlación de intrusiones, el estado del arte y una taxonomía en función del comportamiento de los sistemas.

El capítulo 4 describe el sistema de correlación de alertas propuesto en este trabajo. Se resalta la necesidad de la complementación de los NIDS, y se indica detalladamente el proceso de construcción de cada criterio de análisis aplicado.

El capítulo 5 contiene los resultados de las pruebas locales, y los resultados obtenidos al analizar las trazas de tráfico cedidas por el Centro de Cálculo de la Universidad Complutense de Madrid, capturadas a lo largo del año 2011.

Por último, el capítulo 6 muestra las principales conclusiones extraídas de este trabajo así como algunas líneas futuras de trabajo.

Capítulo 2

Sistemas Detectores de Intrusiones

En este capítulo se introduce el concepto de Sistema de Detección de Intrusiones conocido como IDS. Para ello se establecen sus arquitecturas más representativas y las técnicas que han sido utilizadas a lo largo de los años como método de detección de intrusos, así como los temas de investigación más candentes de la actualidad en el área de la detección de intrusiones. Por último se presenta una taxonomía de los NIDS basados en anomalías en función de las estrategias de modelado del tráfico de la red.

2.1. Estructura general de un IDS

A finales de los años 90, la agencia norteamericana DARPA (*Defense Advanced Research Projects Agency*), constituyó un grupo de investigación enfocado en el desarrollo de una propuesta de plataforma para la detección de intrusiones conocida como CIDF (*Common Intrusion Detection Framework*) [25]. En ella se plantea que los componentes del IDS pueden encontrarse en una misma máquina o distribuidos entre diferentes equipos. A pesar de ser una de los propuestas pioneras, la arquitectura CIDF ha dado pie a la estandarización del IDS en el año 2006 [26].

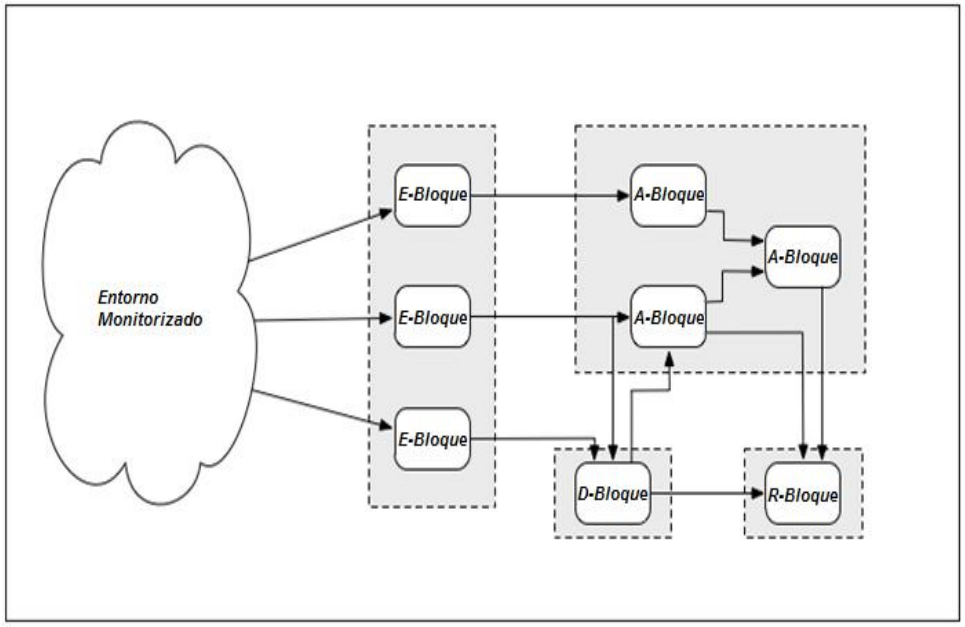

Figura 2.1: Esquema de la arquitectura CIDF

La Figura 2.1 representa el esquema básico de la arquitectura CIDF contenida en el estándar del año 2006. Cabe mencionar que la división entre los distintos componentes es funcional y no física. Esto quiere decir que los módulos pueden estar situados en distintas máquinas con representaciones internas de eventos muy dispares entre sí, por lo que se hace necesario el uso de un formato independiente de comunicación entre ellos. Esta propuesta ha dado pie al lenguaje CISL (*Common Intrusion Specification Language*) que tenía entre sus metas: la expresividad para definir cualquier tipo de intrusión, el ser unívoco en caracterizar intrusiones, la flexibilidad para dar cabida a nuevas intrusiones, la simplicidad en la construcción de las representaciones y la portabilidad para poder ser implementado sobre una amplia variedad de plataformas [27].

A continuación se describe brevemente los distintos módulos que componen la arquitectura CIDF.

- **Entorno:** El entorno monitorizado va a repercutir en la fase de diseño del resto de módulos. Ha de considerarse, que no es los mismo auditar los eventos producidos en una red, que los eventos producidos en un sistema operativo.
- **E-bloque:** Los bloques de escucha proporcionan información sobre los eventos del entorno al resto de elementos del IDS. Esta información se materializa en

forma de objetos de comunicación GIDO (*General Intrusion Detection Object*).

- **A-bloque:** Los bloques del motor de análisis se encargan de analizar los datos recogidos por los bloques de escucha y clasificarlos. Los bloques de análisis reciben y envían objetos de comunicación, los objetos enviados por un módulo de análisis se presupone que sintetizan los eventos de entrada.

- **D-bloque:** Los bloques de la base de datos son los encargados de almacenar objetos GIDO observados en los E-blocks para su procesamiento en los bloques de análisis y de respuesta si esto fuera necesario.

- **R-bloque:** Los bloques del motor de respuesta consumen objetos GIDO. La respuesta puede ser pasiva o activa lo cual significa que de manera automática realiza acciones contra la intrusión. El empleo de respuestas activas ha de ser cuidadosamente considerado ya que pueden incurrir ante un falso positivo en una denegación de servicio a un usuario legítimo.

En el 2001 el CIDF se fusionó con el IETF (*Internet Engineering Task Force*) para pasar a denominarse IDWG (*Intrusión Detection Working Group*) que en el año 2007 publicaron el formato estándar para el intercambio de mensajes entre IDS conocido como IDMEF (*Intrusión Detection Message Exchange Format*) [1].

Pese a que el CIDF no tuvo éxito como estándar para el desarrollo de nuevos sistemas de detección de intrusos resulta satisfactorio como ejemplo genérico de los componentes de cualquier IDS y de las relaciones existentes entre ellos. De esta manera fue considerado como la arquitectura básica de un IDS acorde al estandar de la ISO. Cabe destacar que en la actualidad es cada vez más necesaria la colaboración entre distintos IDS para la correcta detección de las amenazas y la minimización de las falsas alarmas por lo que cada vez resulta más necesario la actualización del estándar considerando las nuevas tendencias de diseño.

2.2. Clasificación de eventos y evaluación

El IDS clasifica los eventos producidos en el sistema monitorizado, etiquetándolos como positivos y negativos. Ésta clasificación se divide en cuatro subcategorías como se muestra en la figura 2.2 [28].

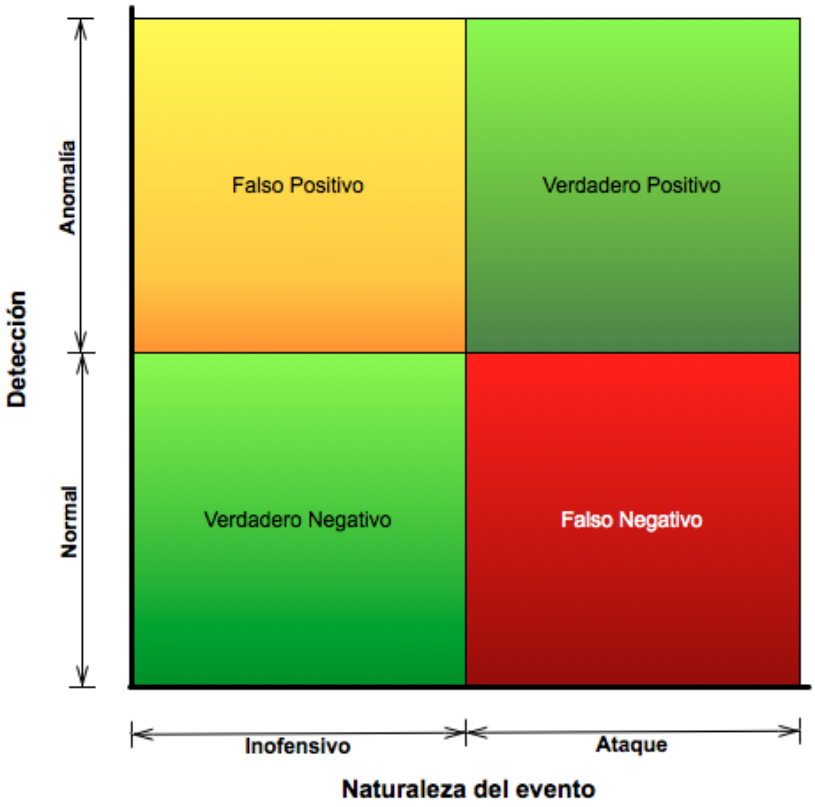

Figura 2.2: Clasificación de los eventos analizados

En ella, los conjuntos de eventos marcados en verde (falsos positivos y verdaderos positivos) son en principio inofensivos, bien por que han sido correctamente identificados como pertenecientes a un ataque o por pertenecer a actividades legítimas. Aquellos eventos que son clasificados de manera errónea como pertenecientes a un ataque, pero que en realidad pertenecen a un uso legítimo del sistema, son denominados falsos positivos y están resaltados en naranja. Este tipo de eventos no son deseables, pero en sí mismos no suponen una verdadera amenaza. La última categoría, resaltada en rojo, constituye la más crítica de las cuatro, puesto que los eventos de ésta categoría son aquellos que han sido erróneamente clasificados como inofensivos cuando en realidad pertenecen a actividades maliciosas.

El rendimiento de un IDS se calcula en base al porcentaje de falsos positivos y la tasa de verdaderos positivos. A pesar de la ausencia de un marco de evaluación común, existe un gran abanico de herramientas analíticas que pueden ser usadas para evaluar aspectos específicos de los sistemas de detección. Como ejemplo, cabe destacar el uso de las denominadas curvas de ROC (*Receiver Operating Characteristic*). Para su evaluación enfrentan el número de verdaderos positivos de un sistema

frente a la probabilidad de aparición de falsos negativos. El método para obtener estas curvas varía de un sistema a otro, pero usualmente se consiguen modificando alguno de los parámetros del algoritmo empleado para realizar la detección.

2.2.1. Conjuntos de datos para la evaluación de NIDS

A la hora evaluar el comportamiento de un NIDS, ha de considerarse trazas de tráfico representativas y cuya naturaleza haya sido validada. En la actualidad, los conjuntos de trazas de tráfico de DARPA [29] [30] constituyen un estándar funcional.

Las trazas de tráfico DARPA han sido utilizadas como método de evaluación en multitud de sistemas de detección de intrusiones basados en redes. Son conjuntos de datos que se usan para comparar los resultados arrojados por las diferentes propuestas. Han sido publicados dos conjuntos de datos, los del año 1998 y los del año 1999. Inicialmente fueron diseñados para evaluar la tasa de falsas alarmas y la tasa de detección de los diferentes NIDS. Los datos fueron recogidos de una red local que simulaba una base de las fuerzas aéreas norteamericanas y contenía tanto ataques conocidos como nuevos.

El conjunto de datos del 1998 estaba compuesto por 32 tipos de ataque, recogidos durante nueve semanas de actividad (siete semanas con el fin de entrenar los algoritmos propuestos, y otras dos semanas de datos para las pruebas). En el conjunto del 1999 se lanzaron más de 300 ataques de 38 tipos, en la red que simulaba cientos de usuarios en miles de estaciones. A pesar de las críticas recibidas [31], su utilización ha perdurado, y siguen siendo una prueba referente a la hora de validar los NIDS actuales.

2.3. Clasificaciones generales de los IDS

Desde su concepción en los años 80 hasta hoy, ha surgido una gran variedad de propuestas de IDS. Es posible encontrar las principales motivaciones de este hecho en la rápida evolución, la sofisticación y la variedad de los ataques así como en la disminución de las habilidades requeridas para llevarlos a cabo.

Debido a la enorme cantidad de propuestas, se hace necesario escoger una serie de ejes (características definitorias) en torno a los que agrupar la taxonomía con el

fin de poder establecer una clasificación correcta para cualquier IDS existente. A continuación se presentan los ejes más importantes para clasificar IDS actuales (estrategia de respuesta, entorno de monitorización, estrategia de detección y tćnicas de análisis).

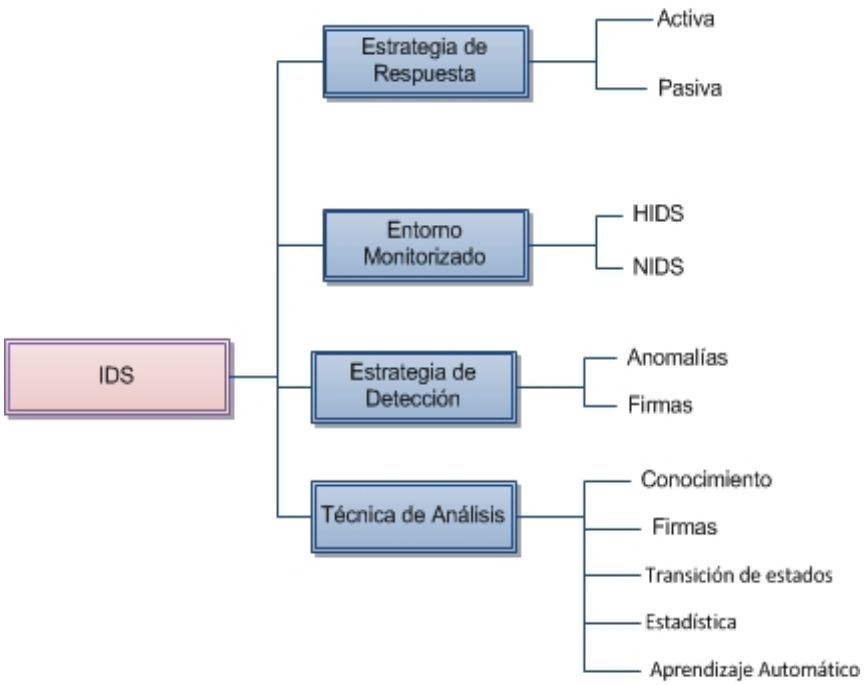

Figura 2.3: Clasificación general de los IDS

La figura 2.3 muestra las diferentes categorías propuestas en la taxonomía. A continuación se explica detalladamente cada una de ellas.

2.3.1. Clasificación por estrategia de respuesta

Cuando el detector etiqueta un evento como malicioso, puede llevar a cabo dos tipos de acciones: Por un lado, si se trata de un IDS con respuesta pasiva, tan sólo se avisará al supervisor del incidente. Pero si el IDS emplea una respuesta activa, además ejecutará contramedidas para intentar ralentizar e incluso detener un intento de intrusión.

Típicamente, los IDS presentan respuesta pasiva, siendo la respuesta activa una característica propia de los IPS (*Intrusion Prevention Systems*). Este tipo de respuesta consiste en enviar alertas al administrador del sistema cuando un evento

supera el umbral de confianza especificado para el detector. El principal problema de este tipo de IDS es que puede sobrepasar a los operadores humanos con alertas procedentes de un mismo ataque, o simplemente con la generación de falsos positivos. En estos casos es necesario complementar el IDS con mecanismos para la correlación de alertas

Por su parte, es de fácil comprensión que la respuesta activa presenta ventajas fundamentales en la protección de cualquier sistema, puesto que una pronta respuesta ante un ataque significa que la integridad del sistema estará mejor protegida que cuando se ha de esperar a que el operador humano analice las alertas generadas por el IDS. No obstante, la automatización de los procesos preventivos no siempre es posible, especialmente cuando se trata con ataques desconocidos. Como se explica posteriormente, también implica riesgos adicionales.

La naturaleza de las acciones que pueden ser llevadas a cabo por parte de un IDS en respuesta a un positivo en el análisis, dependen esencialmente de si se trata de un NIDS (*Network-based Intrusion Detection System*) o de un HIDS (*Host-based Intrusion Detection System*) [32].

Contramedidas de red

Existen cuatro clases de contramedidas que se pueden utilizar desde la perspectiva de un NIDS. Estas medidas corresponden a las capas de la pila de protocolos TCP/IP, quedándo excluidas por motivos obvios las de la capa física.

1 *Contramedidas de la capa de enlace:* La desconexión de un puerto asociado a un sistema desde el que se está lanzando un ataque. Esto es únicamente factible si el ataque está siendo lanzado desde un sistema dentro de la red local.

2 *Contramedidas de la capa de red:* Interactuar con el cortafuegos externo o con las tablas del enrutador para bloquear todas las comunicaciones procedentes de una IP concreta.

3 *Contramedidas de la capa de transporte:* Se generan paquetes TCP RST para deshabilitar la sesión TCP desde la que procede el ataque. De la misma manera se enviaran paquetes ICMP de error si el ataque se está realizando mediante el protocolo UDP

4 *Contramedidas de la capa de aplicación:* Éstas medidas solo son aplicables si el IDS dispone de modo *in-line* y éste se encuentra habilitado. Ésto significa que está realizando el análisis sobre datos reales y no sobre copias de los mismos. Las contramedidas de la capa de aplicación consisten en alterar las cargas útiles maliciosas para convertirlas en inofensivas antes de alcanzar su objetivo. El proceso además requiere modificar los códigos de comprobación de error correspondientes a estos paquetes.

Las contramedidas de la capa de enlace y de la capa de aplicación requieren de mecanismos de tiempo de espera (*timeout*) para volver a habilitar los puertos y las direcciones IP bloqueadas, mientras que los de las capas de transporte y aplicación no los precisan debido a que están vinculados a las sesiones TCP.

Contramedidas locales

La principal ventaja de implementar contramedidas locales es el acceso a las interfaces de programación de aplicaciones conocidas como API (*Application Programming Interface*) y al núcleo mismo del sistema operativo (SO) hace inútiles la mayoría de los intentos de intrusión. Normalmente cuando una aplicación se encuentra comprometida realizará alguna llamada a la API del núcleo del sistema operativo. Las respuestas posibles tras la detección de cualquiera de estas acciones se dividen en dos categorías: modificaciones de sistema y cuñas de aplicación, aunque es común que el propio sistema operativo disponga de medidas preventivas.

1. *Modificación del sistema:* Este conjunto de técnicas incluye la alteración de permisos en el sistema de ficheros, borrado de virus y gusanos, modificación del conjunto de reglas del cortafuegos local.

2. *Cuñas de aplicación:* Se trata de código especifico que intercepta las interacciones de una aplicación con el SO y las inspecciona para detectar código malicioso. Para el SO la cuña no es más que otra aplicación y se encuentra sometida a las mismas restricciones de seguridad que el resto de aplicaciones del sistema.

3. *Mecanismos proporcionados por el SO:* Es conveniente que el sistema operativo protegido implemente mecanismos a nivel de núcleo para reforzar cualquier posible contramedida tomada por el IDS. Ejemplos de estos mecanismos son

el LIDS (*Linux Intrusion Dtetection System*) y el MAC (*Mandatory Access Control*) también para sistemas GNU-Linux y de los cuales se puede encontrar una implementación de referencia en [33].

Pese a constituir una mejora respecto a la simple observación de un registro de alertas, la respuesta activa exacerba el problema de los falsos positivos. En un IDS con respuesta pasiva, los falsos positivos pueden llegar a ser muy molestos, puesto que pueden llegar a sobrepasar la capacidad de los operadores humanos para discriminar entre una alerta falsa y una real [34], sin embargo cuando se emplea la respuesta activa es posible llegar a incurrir en denegaciones de servicio arbitrarias a usuarios legítimos, por lo que el rango de circunstancias en las que se emplea una respuesta activa ha de ser limitado a un pequeño conjunto de actividades maliciosas bien definidas.

2.3.2. Clasificación por entorno monitorizado

Un IDS puede monitorizar eventos locales (HIDS) o eventos en una red (NIDS). A continuación se explica detalladamente las ideas principales de cada una de ellas.

Sistemas de Detección de Intrusiones basados en *Hosts* (HIDS)

Este tipo de IDS fue el primero en ser desarrollado y desplegado. Típicamente están instalados sobre el servidor principal, elementos claves o elementos expuestos del sistema al que brinda protección. Los HIDS operan monitorizando cambios en los registros de auditoría del *host* tales como: procesos del sistema, uso de CPU, acceso a ficheros, cuentas de usuario, políticas de auditoria o registro de eventos. Si los cambios en alguno de estos indicadores excede el umbral de confianza del HIDS o hace peligrar la integridad del sistema protegido, el HIDS lleva a cabo una acción a modo de respuesta.

La principal ventaja de estos detectores es su relativo bajo coste computacional, puesto que monitorizan información de alto nivel. Sin embargo, esto también constituye su principal inconveniente ya que sólo pueden lanzar alertas una vez que el daño al sistema está hecho. la Figura 2.4 puede verse el esquema de un sistema protegido por un HIDS.

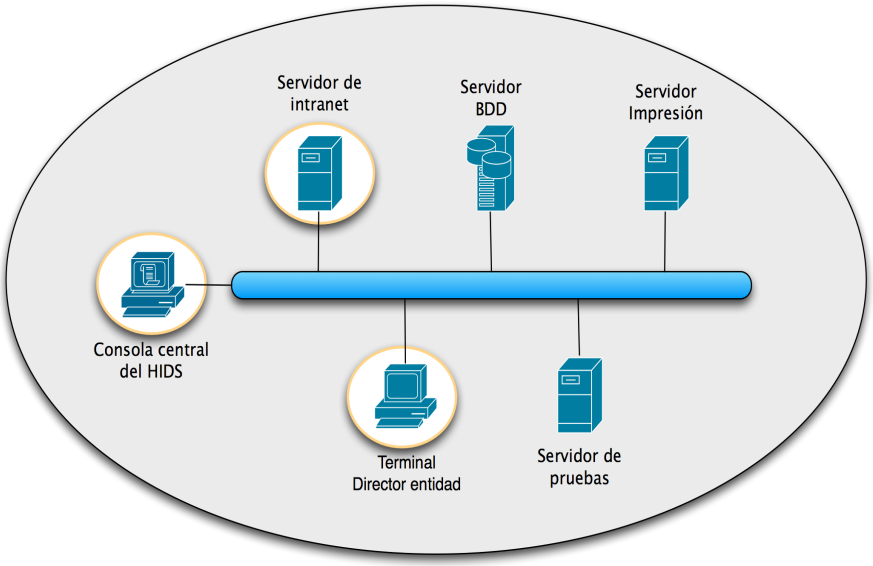

Figura 2.4: Ejemplo de sistema protegido mediante HIDS

Sistemas de Detección de Intrusiones basados en Redes (NIDS)

Los NIDS analizan eventos provenientes del tráfico de red. Existen diversas formas de clasificar los NIDS: por objeto de análisis, por situación de los componentes del IDS, por tipo de análisis efectuado y por último por la técnica de análisis empleada. Para el propósito de este documento tan sólo se entra en detalle sobre el objeto de análisis.

Los objetos del análisis de la red son los paquetes y tramas de los diversos protocolos de la misma. Estos componentes del tráfico de la red se dividen en dos partes: cabecera y contenedor. En consecuencia se han desarrollado NIDS que abordan el problema de la detección de intrusiones mediante el análisis de una de estos componentes. En la figura 2.5 puede apreciarse un ejemplo de sistema protegido mediante un NIDS monitorizando un segmento de red.

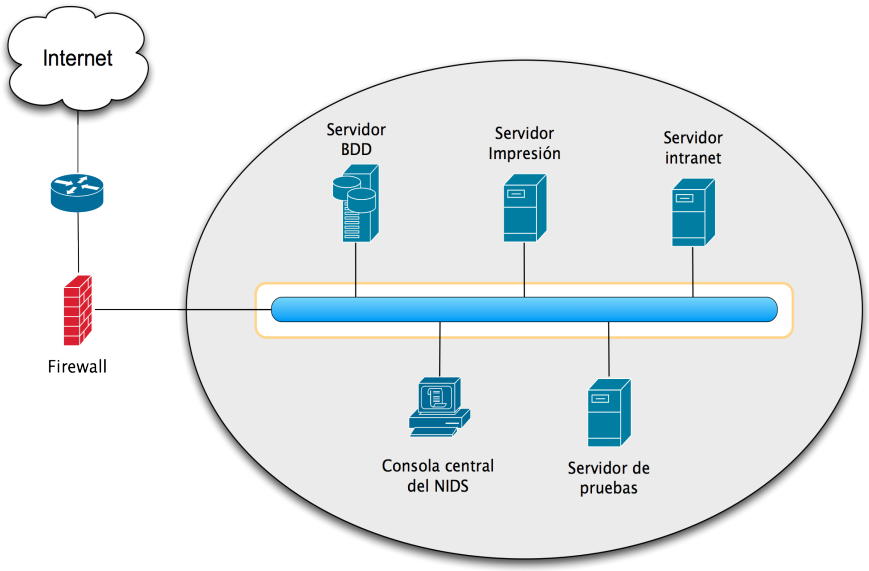

Figura 2.5: Ejemplo de sistema protegido mediante NIDS

Análisis de cabeceras

El análisis de cabeceras se basa en analizar el flujo de paquetes teniendo en cuenta los datos de alto nivel de las cabeceras de los mismos. Este análisis presenta un buen rendimiento en la detección de ataques de desbordamiento, en los que se envían una gran cantidad de paquetes. Se intentan descubrir patrones con respecto al puerto destino, protocolo utilizado y otra información presente en la cabecera. Esto permite analizar gran número de paquetes a un coste computacional relativamente bajo. El problema para esta aproximación se da cuando se intentan descubrir ataques en los cuales la información distintiva respecto al tráfico legítimo se encuentra en la carga útil y no en las cabeceras.

Otro punto a tratar en este tipo de detectores es el siguiente: una de las informaciones más utilizadas por estos analizadores es el puerto de destino del paquete, siendo esto utilizado para determinar el protocolo que está utilizando el servicio. Sin embargo esto presenta una limitación importante, ya que frecuentemente las aplicaciones hacen uso de puertos no estándar, siendo un ejemplo de esto, las aplicaciones tipo VoIP (*Voice over IP*) como Skype, la cual lleva a cabo dicha acción para permitir las comunicaciones sin interferencias de los cortafuegos locales. Éste mismo comportamiento es explotado por parte de los atacantes con los mismos fines, siendo necesario la implementación de otros mecanismos a nivel de aplicación para determinar el protocolo utilizado.

Análisis de la carga útil

Al analizar la carga útil, al contrario del análisis de cabeceras se procesan datos de bajo nivel que se encuentran dentro del paquete capturado de la red monitorizada. De nuevo es posible encontrar trabajos centrados en analizar la carga útil de distintas maneras dependiendo del protocolo al que pertenezca el paquete. Sin embargo, esto presenta un problema en la adaptación a nuevos servicios.

En otro sentido se han propuesto alternativas que analizan la carga útil independientemente del protocolo utilizado, siendo estas más costosas pero presentando una mejor adaptación ante nuevas circunstancias. El análisis de la carga útil ha demostrado ser útil en la detección de ataques compuestos de un sólo paquete. Ejemplos de este tipo de ataques son los conocidos U2R(*User To Root*) o el R2R(*Remote To Root*).

El principal problema que presenta este análisis es el coste en términos de recursos para el sistema que efectúa el análisis, debido a la vectorización de los paquetes para su procesamiento. Otra dificultad añadida es la heterogeneidad en la carga útil del tráfico que circulan en la red.

Consideraciones sobre el uso de NIDS

Un NIDS actúa como un *sniffer* del trafico de red, capturando y descodificando paquetes que circulen por el segmento de red sobre el que ha sido desplegado. A pesar de su indudable valor como elemento de alerta temprana, hay varios factores a considerar previos a su despliegue.

En redes conmutadas: en una red conmutada, las comunicaciones se establecen mediante un circuito propietario en lugar de por un canal común, como en una red convencional. Para solucionar esto se ha de configurar el NIDS para que escuche el puerto de monitorización de la red.

En redes de alta velocidad: cuando se despliega un NIDS la velocidad de análisis es un factor muy importante a tener en cuenta. Un IDS sin la suficiente velocidad de procesamiento no analizará todo el tráfico de la red. Esto puede ser usado por un atacante, el cual sobrecargará la red para poder introducir *exploits* que no serán detectados. Para solucionar esto, los diferentes proveedores ofrecen soluciones con *hardware* dedicado para mejorar el rendimiento. Otra posible solución es disminuir la velocidad de la red para adecuarla a la velocidad de proceso del IDS.

Redes codificadas: si el atacante hace uso de una conexión codificada, el NIDS no será capaz de detectar el ataque, a no ser que sea capaz de descifrar la carga útil. En este caso un HIDS deberá de ser emplazado en la máquina que ofrece la conexión cifrada para vigilar su comportamiento.

2.3.3. Clasificación por estrategia de detección

La estrategia de detección se refiere a la aproximación elegida para distinguir el uso legítimo y el uso anómalo del sistema. La mayor parte de autores coinciden al identificar dos categorías principales y una derivada de ellas: IDS basados en detección de firmas, IDS basados en detección de anomalías y por último IDS híbridos.

Detección basada en firmas

Los IDS que basan su estrategia de detección de firmas monitorizan el uso indebido de los recursos y realizan su trabajo mediante la clasificación de los eventos de acuerdo a una base de conocimiento que guarda firmas de ataques conocidos. Si una secuencia de eventos del sistema coincide con una firma almacenada en la base de datos se disparará una alarma.

Originalmente las firmas eran construidas a mano usando en conocimiento de expertos. Sin embargo hoy en día este proceso se ha agilizado gracias a la utilización de técnicas de minería de datos sobre registros de intrusiones conocidas [35]. Estos sistemas funcionan de manera muy precisa siempre que se trate de patrones de ataques conocidos, sin embargo resulta bastante fácil engañarlos con la introducción de variaciones en los eventos de los ataques, y resultan inútiles frente a ataques desconocidos [36].

Detección basada en anomalías

Por el contrario, los NIDS orientados a la detección de anomalías se basan en el modelado del modo de uso habitual y legítimo del sistema y en la detección de desviaciones sobre el misma. Han recibido una gran atención en los últimos años debido a que tienen la interesante propiedad de poder hacer frente a amenazas de nueva aparición. Sin embargo presentan varios problemas relacionados tanto con su proceso de construcción como con el elevado ratio de falsos positivos que suelen arrojar. Estos inconvenientes se explican en una sección posterior más detalladamente.

Detección híbrida

Existe también la posibilidad de combinar ambas estrategias dando lugar a detectores híbridos, los cuales intentan combinar los puntos fuertes de las estrategias antes descritas. Dado que los detectores de anomalías no constituyen una respuesta completa al problema de la detección de intrusiones, usualmente han de ser combinados con estrategias de detección de uso indebido. Las razones para esto son el alto índice de aciertos y el bajo número de falsos positivos ante vulnerabilidades conocidas de los IDS basados en uso indebido, y la protección frente a nuevos ataques de los IDS basados en la de detección de anomalías. Un ejemplo de este tipo de IDS es EMERALD [37], que realiza ambos tipos de análisis. Otra característica interesante de EMERALD es que hace uso de la plataforma CIDF para permitir la comunicación entre sus elementos de análisis y la adición de módulos procedentes de terceras partes permitiendo así configurar el sistema de detección para responder a las necesidades de la organización que lo despliegue.

2.3.4. Clasificación por técnicas de análisis

También se pueden clasificar en función de la técnica elegida para implementar el clasificador de eventos en los módulos de análisis del IDS. Existen 5 grandes estrategias: basadas en el conocimiento, basadas en el análisis de firmas, basadas en análisis de la transición de estados, basadas en métodos estadísticos y por último aquellas basadas en el aprendizaje automático.

Sistemas basados en el conocimiento

Basados en reglas de tipo *if-then-else*, si algún evento captado del entorno satisface todas las precondiciones de una regla, esta se disparará. Ofrecen la ventaja de separar la lógica de control del dominio del problema, sin embargo no son una buena alternativa para analizar secuencias de eventos. Ejemplo de este tipo de IDS es P-Best, aunque este sea un intérprete utilizado por otros IDS como [36] [37].

Análisis de firmas

Observa la ocurrencia de cadenas especiales en los eventos monitorizados por el IDS y los compara con los almacenados en una base de datos con firmas de ataques. El principal problema de este enfoque es la necesidad de incorporar una nueva firma a la base de datos por cada nuevo ataque conocido. Quizás el ejemplo más conocido de estos IDS sea Snort [38], debido a su licencia libre y capacidad de desarrollo de preprocesadores y reglas por parte de la comunidad.

Análisis de transición de estados
Se trata de autómatas finitos que a través de transiciones basadas en los estados de seguridad del sistema, intentan modelar el proceso de una intrusión. Cuando el autómata alcanza un estado considerado de peligro lanza una alerta. Las representantes de estos sistemas que alcanzaron mayor difusión fueron NetSTAT y USTAT (basados en red y local respectivamente) [39].

Sistemas basados en métodos estadísticos
Se basan en modelizar el comportamiento de los usuarios en base a métricas extraídas (estrategia de detección de anomalías) de los registros de auditorías del sistema. Una característica clásica de estos sistemas es el establecimiento de perfiles para los distintos usuarios del sistema [40].

El proceso de comparación consiste en cotejar un vector de variables que han sido recogidas del sistema, con otro almacenado que guarda el comportamiento normal de ese usuario. Si la desviación supera un valor predefinido, se lanzará una alarma. Es conveniente actualizar los perfiles cada cierto tiempo, usando para ello los registros de actividad del usuario más recientes, para de esa manera moderar el número de falsos positivos de la herramienta. Sin embargo ésta es una fuente de vulnerabilidad de estos sistemas, puesto que un atacante puede desentrenar la herramienta para que permita una intrusión. Un ejemplo de estos sistemas es el ISA-IDS [39].

Sistemas basados en aprendizaje automático: Se trata de sistemas que realizan uso de técnicas que no necesitan intervención humana para extraer los modelos que representan o bien ataques, o bien el uso legítimo del sistema. La extracción de los modelos se realiza durante la fase de entrenamiento. Ejemplos de conjuntos representativos disponibles en la red son los elaborados por la DARPA en 1998 y 1999 así como el proveniente de la KDD Cup de 1999 organizada por la ACM. También pueden ser entrenados con datos recogidos de redes reales. Con respecto al mecanismo de entrenamiento se puede hacer la división entre entrenamiento supervisado y no supervisado.

El aprendizaje supervisado hace uso de conjuntos etiquetados, teniendo como desventaja la escasez de dichos conjuntos de datos, ya que pocas organizaciones guardan conjuntos de datos exhaustivamente etiquetados. Por el contrario, el aprendizaje no supervisado no precisa de conjuntos etiquetados. Muchas técnicas de aprendizaje

automático pueden hacer uso de ambos métodos en distintas etapas del entrenamiento. Por ejemplo [41] hace uso de ambas.

Estos sistemas han recibido gran atención tanto en el desarrollo de IDS orientados a detección de anomalías como en el de uso indebido. Ésto se debe a que permiten la extracción de conocimiento sobre el dominio del problema, minimizando la intervención humana y el tiempo que se tarda en preparar las defensas frente a nuevas amenazas.

2.4. Clasificación de IDS basados en anomalías en base al método de modelado

Dado que el presente trabajo se centra el la correlación de las alertas emitidas por NIDS basados en anomalías, resulta interesante hacer hincapié en este tipo de sistemas. La técnica de modelado que utilice el NIDS dependerá del tipo de anomalía a las que se tenga que enfrentar, del tipo y comportamiento de los datos, del entorno en el que el sistema esté funcionando, de las limitaciones de coste o del cálculo, y del nivel de seguridad requerido.

Al realizar la implantación de un IDS se debe tener en cuenta que un buen entrenamiento es de vital importancia en la efectividad del sistema. Asimismo, el modelo utilizado debe reflejar lo más fielmente posible el comportamiento del sistema en ausencia de ataques, para lo cual se debe contar con un tráfico lo más limpio posible. El entrenamiento puede definirse como lo suficientemente largo hasta que se construya un modelo completo del entorno de aplicación, pero ante una actualización del sistema, el repetir esta fase puede tener un sobrecoste mayor. Si la duración de esta fase es muy corta puede darse una inadecuada clasificación, viéndose incrementadas en la fase de detección las alertas de tráfico legítimo señalizado como anómalo (falsos positivos).

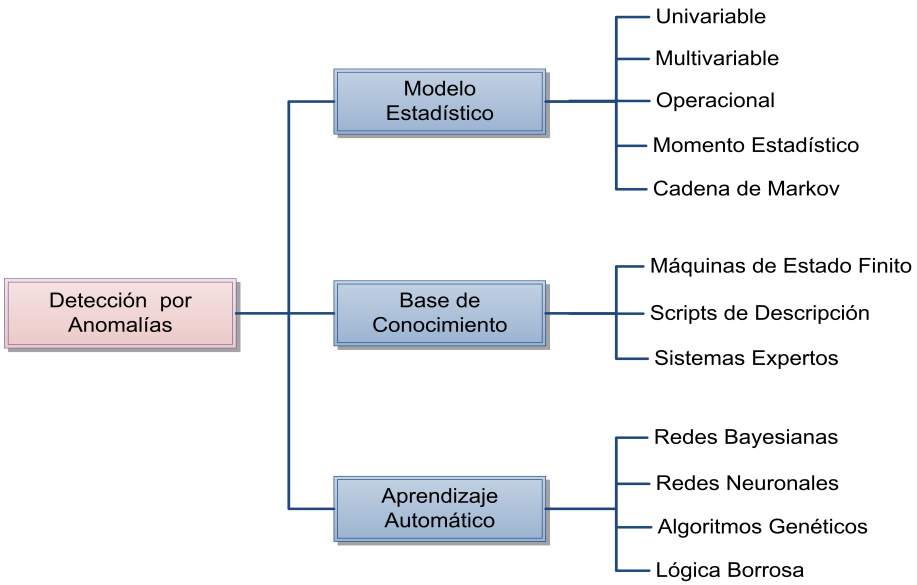

Figura 2.6: Estrategias de modelado

En la figura 2.6 se presenta una clasificación de los NIDS basados en anomalías de acuerdo al comportamiento del modelo de procesamiento de los datos [42] identificando las siguientes categorías: modelos estadísticos, entre los cuales se pueden mencionar el operacional, las cadenas de Markov, el modelo multivariante, modelos basados en el momento estadístico y modelos univariantes. Por otro lado existen estrategias orientadas al uso de bases del conocimiento, como por ejemplo los sistemas expertos, las máquinas de estados finitas y el uso de *scripts* de descripción; También pueden adoptar estrategias de Aprendizaje Automático como el uso de Redes Bayesianas, algoritmos genéticos, redes neuronales artificiales o la lógica borrosa.

A continuación se explica en profundidad cada uno de ellos.

2.4.1. Modelos estadísticos

Gran cantidad de los NIDS actuales han optado por el uso de modelos estadísticos. Dichos sistemas tienen como objetivo la construcción de modelos estadístico-predictivos capaces de identificar el uso anómalo del sistema frente al uso legítimo. Mediante esta técnica se captura el tráfico de red y se crea un perfil que represente su comportamiento estocástico. Este perfil utiliza métricas como la tasa de tráfico, el número de paquetes para cada protocolo, la tasa de conexiones o el número de direcciones IP diferentes, entre otras, que permiten representar los diferentes modos de empleo del sistema. En el proceso de detección se consideran dos conjuntos de tráfico de red. Uno de ellos representa las características observadas en el momento

del análisis, mientras que el otro representa las características conocidas previamente.

La clasificación se produce en función del grado de similitud entre ambos conjuntos, etiquetando como anomalía al tráfico que difiera considerablemente del tráfico legítimo. Neumann et al. [37] introducen en su sistema EMERALD el uso de metodologías distribuidas de correlación de grandes cantidades de eventos. Para ello combinan el análisis de firmas con perfiles estadísticos que permiten llevar a cabo clasificaciones de tráfico en tiempo real para cualquiera de los servicios de que entonces disponían las redes, marcando un hito en el campo de los NIDS basados en anomalías y en modelos estadísticos. Años más tarde, Sang et al. [43] proponen el primer método de detección de anomalías mediante el uso de algoritmos de agrupamiento de información conocidos como *clustering*.

A partir de estos primeros trabajos se comienza a aplicar diferentes herramientas estadísticas. Así, Yu et al. [44] presentan un enfoque de detección de anomalías basado en un modelado adaptativo no-paramétrico en el tráfico de redes simétricas, que tiene la capacidad de ajustar sus parámetros de detección a la situación en la que se encuentre la red en que opera. Previamente, Ye et al. [45] ya habían aprovechado la robustez del modelo de Markov para la clasificación de eventos, aunque sólo lograron buenos resultados con datos poco distorsionados.

Una de las ventajas del diseño basado en el modelado estadístico es que no requiere de conjuntos de entrenamiento de ataques conocidos para el proceso de generación del modelo. Además la mayoría de estas propuestas han sido bastante buenas en condiciones de tráfico real [46], no causando apenas sobrecarga en la red. Sin embargo, cabe anotar que al aplicar estas herramientas se asume que el comportamiento del tráfico de las redes es cuasi-estacionario, algo que no siempre se puede garantizar.

2.4.2. Modelos basados en conocimiento

Una base de conocimientos es un tipo de base de datos adaptada a la gestión y representación del conocimiento. Los NIDS que incorporan estos mecanismos requieren una fase de entrenamiento capaz de identificar los parámetros más representativos de los conjuntos de tráfico legítimo y malicioso con que se pretendan entrenar. Una vez extraídos, se genera una base de reglas capaz de clasificar la naturaleza del tráfico

analizado. Lee et al. [35] presentan un NIDS con una base de conocimiento que analiza el tráfico en función del contenido de la carga útil de los paquetes considerando las características propias de la conexión. Jiang et al. [47] incorporan un enfoque distribuido de detección de intrusiones basado en máquinas de estado finito, con un esquema de detección basado en agrupamiento, en el que periódicamente se elige un nodo como monitor del agrupamiento. Tran et al. [48]. proponen un marco de clasificación multi-experto para detectar diferentes tipos de anomalías de red a través de técnicas de detección en las que se seleccionan distintos atributos y algoritmos de aprendizaje.

En general, las ventajas más significativas de la utilización de bases de conocimiento en el diseño de NIDS son el alto grado de solidez y de flexibilidad que les otorgan. Sin embargo, el uso de análisis basado en reglas puede sobrecargar la red en la que opera si no se emplean métodos de agregación de reglas. Además, determinados diseños pueden requerir demasiado conocimiento previo de las amenazas a las que se enfrenta, pudiendo aproximarse demasiado a la detección por firmas.

2.4.3. Modelados basados en aprendizaje automático

El uso de este tipo de técnicas permiten al NIDS aprender de eventos conocidos con el fin de llevar a cabo clasificaciones de eventos desconocidos, generalizando sobre los conocimientos que ha adquirido. Consecuentemente, un NIDS basado en anomalías con aprendizaje automático tiene la capacidad de cambiar su estrategia de clasificación mediante la adquisición de nueva información. Precisamente una característica singular de estos esquemas es la necesidad de datos etiquetados para entrenar el modelo de comportamiento, una condición que a veces puede ser un problema, debido a que etiquetados incorrectos pueden conducirle a un comportamiento no deseado. Para evitarlo, se suelen emplear mecanismos de aprendizaje tolerantes a un cierto margen de ruido.

El aprendizaje automático tienen un alto grado de similitud con las estrategias estadísticas anteriormente citadas, pero su enfoque se basa directamente en la optimización del coste computacional de aquellos algoritmos que pueden sobrecargar la red. A pesar de su alto rendimiento, hay trabajos que han seguido otras ramas. Un ejemplo es la propuesta publicada por Song et al. [49], en la cual desarrollan una solución mediante *hardware* reprogramable para lograr una clasificación de paquetes eficiente en un NIDS implementado en una FPGA (*Field Programmable Gate*

Arrays), tecnología ampliamente utilizada para el análisis en tiempo real.

Uno de los mayores esfuerzos en la aplicación de este tipo de técnicas se debe a Mahoney et al. [50] quienes propone tres sistemas de detección de intrusiones: detector de anomalías en la cabecera de los paquetes (PHAD), detector de anomalías en la capa de aplicación (ALAD) y detector de anomalías en el tráfico de la red (NETAD). Cada uno de ellos extrae cierta información del tráfico analizado y genera una clasificación acorde al entrenamiento previamente recibido.

Pero sin duda la estrategia de detección PAYL desarrollada por Wang et al. [51] representa un hito en el área de la detección de intrusiones. Este sistema clasifica el tráfico basándose en 3 características: el puerto, el tamaño del paquete y la dirección del flujo (entrada o salida). Mediante estos 3 parámetros clasifican la carga útil creando una serie de patrones para definir lo que sería un comportamiento normal dentro de cada clase. A raíz de este trabajo, aparecen diversas propuestas, como la de Bolzoni et al. en la que proponen la estrategia POSEIDON [52] con el fin de solucionar determinadas carencias de PAYL a la hora de llevar a cabo técnicas de agrupamiento. Otra importante aportación de POSEIDON es el uso de mapas auto organizativos SOM (Self-Organizing Map), que además de reducir la sobrecarga del NIDS en la red, reduce el número de clases generadas en los procesos de entrenamiento, permitiéndole de esta manera operar con una mayor precisión. Estos dos trabajos son especialmente importantes ya que gran parte de las propuestas actuales de detección de anomalías en la carga útil del tráfico de la red se basan en ellas.

Capítulo 3

Correlación de Intrusiones

La aparición de nuevas amenazas desconocidas, implica una rápida y eficaz adaptación de los mecanismos de defensa diseñados para hacerles frente. Ésto ha dado lugar a una rápida proliferación de los Sistemas de Detección de Intrusiones, y en muchos casos a una especialización muy concreta frente a determinados tipos de amenazas. La principal consecuencia de dicha especialización es la centralización de los esfuerzos en la detección de las amenazas, y por lo tanto, el descuido o delegación de su interpretación a sistemas complementarios, los comúnmente denominados sistemas de correlación de alertas.

En este capítulo se introduce el concepto de sistema de correlación de intrusiones. Para ello se explica detalladamente cada una de las etapas de la correlación de intrusiones y se presenta una taxonomía en función de su comportamiento.

3.1. Esquema para la correlación de intrusiones

Trabajar directamente con las enormes cantidades de alertas generadas por los distintos sensores de los Sistemas de Detección de Intrusiones instalados en un perímetro de defensa, resulta inviable para cualquier tipo de operador. Ésto inutiliza completamente el esfuerzo llevado a cabo por mecanismos de detección, imposibilitando el tomar las medidas apropiadas a tiempo, con el fin de mitigar la amenaza. Este problema ha llevado a plantear la necesidad de disponer de sistemas capaces de fusionar y correlacionar las alertas, con el fin de simplificar la información obtenida. Para ellos cuentan con bases de conocimiento en ocasiones diferentes a las de los propios sensores, que tienen como objetivo ayudar a establecer la prioridad de tratamiento de cada alerta, y extender la información provista por el sensor para poder llevar a cabo estudios forenses de la amenaza mucho más detallados.

En el trabajo de T. Zang et al. [54] se propone un modelo de procesamiento de alertas jerárquico, que toma como datos de entrada las secuencias de alertas generadas por diferentes sensores y las procesa a diferentes módulos. Dichos módulos se encargan de la normalización, la agregación, la verificación y la correlación de las alertas. En este proceso se automatiza el agrupamiento de alertas a bajo nivel y se procede a su correlación, estableciendo los diferentes niveles de prioridad y de riesgo, permitiendo al operador gestionar su tratamiento de una manera eficaz y precisa. Existen muchas variaciones del modelo propuesto, adaptadas a diferentes escenarios. Pero a pesar de ello, sirve como una buena referencia para conocer los módulos que habitualmente componen los sistemas de correlación de intrusiones.

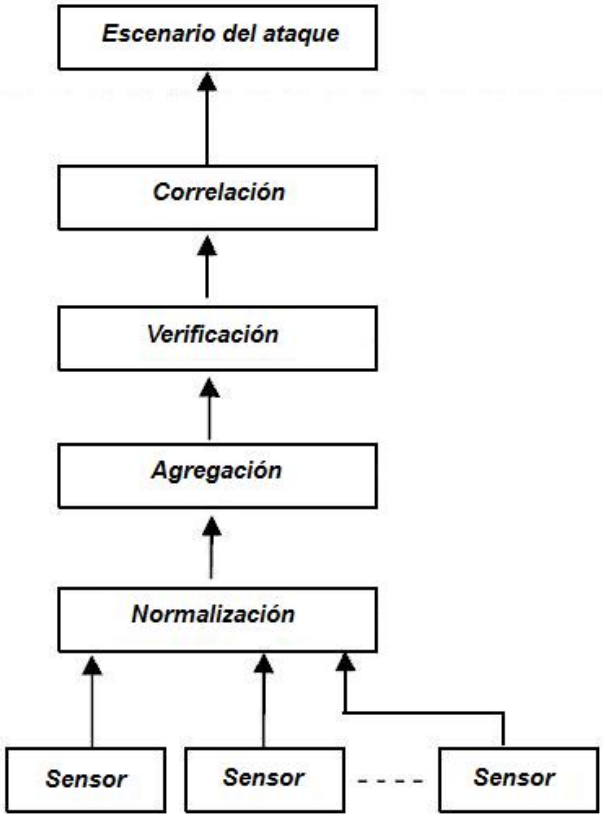

Figura 3.1: Esquema de proceso de correlación de intrusiones de T. Zang

En la figura 3.1 puede verse el esquema de proceso de correlación de intrusiones planteado por T. Zang et al. A continuación se describe brevemente cada uno de sus niveles.

3.1.1. Módulo de normalización de alertas

Los diferentes sensores del perímetro de seguridad emiten alertas en diferentes formatos. El proceso de normalización de alertas tiene como objetivo la conversión de las alertas emitidas por cada uno de ellos en un formato único, conocido por el operador. De esta forma, la manera de interpretar las alertas será la misma en las etapas siguientes del proceso de correlación. En la actualidad, el formato más usado es el estándar IDMEF [1]. Dicho formato corresponde con una gramática del conocido lenguaje de marcado XML (*Extensible Markup Language*). Además de la unificición de alertas, el estándar IDMEF soluciona el problema de la sincronización de los relojes de los sensores que llevan a cabo el procesamiento de los eventos. Para ello utiliza el conocido NTP (*Network Time Protocol*) [55].

3.1.2. Módulo de agregación de alertas

El objetivo de la etapa de agregación de alertas es la agrupación de aquellas alertas generadas con proximidad temporal, y de características similares. Esto va a evitar el procesamiento múltiple de un mismo evento, y la saturación del operador. Por ejemplo, si reincide la aparición de un tipo de alerta producida desde una misma dirección IP, contra un mismo puerto y un mismo servicio en ráfagas temporales, posiblemente se trate de un mismo atacante replicando el ataque. Mediante agregación se unirán todas esas alertas en una única, y se tratarán unísonamente. En éste ámbito han aparecido propuestas muy relevantes, como la de A. Valdes y K. Skinner [3] quienes propusieron en el año 2001 un marco matemático para la correlación de alertas. Otra importante propuesta fue la de S. Staniford et al. [56] orientada a la prevención de escaneos de puertos, y la de K. Julisch [57] para la agrupación de alertas en función de su origen. En base al ejemplo mencionado, H. Debar y A. Wespi [2] propusieron pautas para la correlación en base a la dirección de origen de la amenaza, la dirección destino y la clasificación previa establecida por el IDS.

3.1.3. Módulo de verificación de alertas

La verificación de alertas, analiza cada una de las alertas generadas por los sensores y establece la probabilidad de que el ataque cumpla su propósito con éxito. De esta manera, las alertas asociadas a ataques con menos probabilidades de éxito son marcadas como tales, y por lo tanto su influencia en el proceso de correlación va a ser inferior. Esto conlleva, que si por ejemplo, el sistema protegido corre bajo un sistema operativo GNU-Linux, y se ha detectado el envío de un *malware* que afecta únicamente al sistema operativo Windows, evidentemente la prioridad de

tratamiento y el nivel de riesgo son marcados como muy bajos. Si el sistema protegido funcionara bajo el sistema operativo Windows, entonces la alerta sería marcada como prioritaria, lo que le daría una especial importancia al proceso de prevención. Históricamente se ha usado dos aproximaciones para la verificación de alertas:

- *Verificación pasiva de alertas.* Esta estrategia sólo considera la información provista por los sistemas *host*, la red y sus servicios. Su principal ventaja es que no lleva a cabo comprobaciones extraordinarias que puedan penalizar la calidad de servicio del sistema protegido. El problema es que los cambios en el sistema, pueden confundir al sistema de correlación, y no tiene tanta precisión como la verificación activa. Uno de los trabajos más relevantes en esta área es el de P. Porras et al. [6], en el que se considera la información provista por sistemas de defensa como cortafuegos, HIDS o mecanismos de autenticación. Esto permite ahorrar tiempo de verificación de las alertas, permitiendo una rápida respuesta frente a los ataques.

- *Verificación activa de alertas.* las estrategias de defensa activa necesitan buscar pruebas adicionales del éxito de la amenaza en el sistema contra el que va dirigida. De esta manera, una vez procesada la alerta, el sistema de correlación ejecutará varias pruebas sobre el sistema víctima para comprobar que los efectos nocivos del aviso corresponden con los de la alerta. Es un proceso mucho más lento, pero altamente preciso. Esta aproximación incluye el problema de que un atacante experimentado, conocedor de su funcionamiento, puede intentar ataques de denegación de servicio mediante la reiteración de las comprobaciones.

3.1.4. Módulo de correlación de alertas

La fase de correlación de alertas tiene como objetivo descubrir las similitudes entre el contenido de las alertas que han sido emitidas por los sensores y amenazas conocidas. De esta manera es posible determinar con mayor precisión su naturaleza y el riesgo que implican. De esta manera es posible priorizar la respuesta para mitigarlas. Para ello se emplean diferentes estrategias relacionadas con el campo de la minería de datos, como el aprendizaje automático, técnicas de agrupamiento o el empleo de estrategias de *Soft-Computing* como algoritmos genéticos o lógica difusa.

3.2. Evolución de los sistemas de correlación de alertas

La evolución de los sistemas de correlación de alertas ha sido directamente delimitada por las tendencias en las estrategias de detección de intrusiones. Entre ellas destacan la necesidad del desarrollo de IDS para entornos distribuidos, el cambio del análisis basado en firmas al análisis basado en anomalías, o la necesidad de realizar una correlación eficiente, capaz de procesar las alertas en tiempo real. Éstas son algunas de las motivaciones que han dado lugar a las propuestas que han establecido nuevas pautas de investigación. A continuación se mencionan los trabajos que han guiado su avance hasta las tendencias actuales

3.2.1. ACC

Uno de los primeros trabajos de correlación de alertas fue llevado a cabo por la empresa multinacional IBM. En el año 2001 H. Debat y A. Wespi [2] publicaron la propuesta ACC (*Aggregation and Correlation Component*) para la gestión de las alertas emitidas por múltiples sensores. El objetivo del trabajo era la mitigación del problema de desbordamiento de alertas, la consideración del contexto de las alertas a la hora de llevar a cabo el agrupamiento, la identificación de los fasos positivos emitidos por los sensores y el garantizar la escalabilidad de conjuntos de IDS que cooperan de forma distribuida. Dicho sistema fue desplegado sobre la herramienta para el diagnóstico y resolución de problemas soportada por IBM, denominada TEC (*Tivoli Enterprise Console*).

ACC constaba de tres fases: una fase de normalización de alertas, una fase de correlación en la que se construye y clasifica el escenario del ataque, y una última fase de agregación en base a los escenarios creados. A pesar de que las reglas que presentaron y su estrategia de normalización no eran demasiado robustas, sirvió para resaltar la necesidad de solucionar los problemas planteados. Además fue una de las primeras propuestas de correlación basada en la reconstrucción de escenarios de ataques.

3.2.2. El marco matemátivo de Valdes y Skinner

En el año 2000, A. Valdes y K. Skinner presentaron una propuesta de Sistema de Detección de Intrusiones [4] capaz de analizar el tráfico TCP entrante al sistema protegido. Para ello aplicaron redes bayesianas, y fueron capaces de detectar la ma-

yor parte de los ataques con que fue probado. El sistema resultaba potencialmente escalable a arquitecturas distribuidas, pero los experimentos mostraron su debilidad contra ataques de denegación de servicio a través de la inyección de falsas alertas. Como solución a dicho problema, un año más tarde presentaron un marco matemático para la correlación de alertas basado en establecer la similitud entre las alertas emitidas por diferentes sensores [3].

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	1	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.6
2	0.3	1	0.6	0.3	0.6	0.6	0.6	0.6	0.4	0.3	0.4	0.1	0.5	0.6
3	0.3	0.6	1	0.3	0.6	0.5	0.5	0.4	0.3	0.6	0.4	0.1	0.5	0.6
4	0.3	0.3	0.3	1	0.6	0.3	0.3	0.4	0.5	0.6	0.4	0.1	0.5	0.6
5	0.3	0.2	0.2	0.3	1	0.7	0.3	0.3	0.3	0.3	0.4	0.8	0.3	0.6
6	0.3	0.6	0.5	0.3	0.6	1	0.6	0.6	0.3	0.3	0.4	0.1	0.5	0.6
7	0.3	0.5	0.3	0.5	0.6	0.8	1	0.6	0.5	0.3	0.4	0.1	0.5	0.6
8	0.3	0.5	0.3	0.5	0.6	0.6	0.6	1	0.6	0.3	0.4	0.1	0.5	0.6
9	0.3	0.5	0.5	0.3	0.6	0.6	0.6	0.6	1	0.3	0.4	0.1	0.5	0.6
10	0.3	0.3	0.3	0.6	0.3	0.3	0.3	0.3	0.3	1	0.4	0.4	0.3	0.6
11	0.3	0.3	0.5	0.3	0.5	0.6	0.5	0.6	0.5	0.3	1	0.1	0.3	0.6
12	0.3	0.1	0.1	0.3	0.8	0.3	0.3	0.3	0.3	0.5	0.4	1	0.3	0.6
13	0.3	0.3	0.3	0.3	0.3	0.6	0.3	0.6	0.5	0.5	0.4	0.1	1	0.6
14	0.3	0.3	0.3	0.3	0.6	0.5	0.3	0.3	0.3	0.3	0.4	0.3	0.3	1

Tabla 3.1: Ejemplo de similitud de alertas según el marco matemático

Categoría	**Etiqueta**
NO VÁLIDO	1
VIOLACIÓN DE PRIVILEGIOS	2
USUARIO INESTABLE	3
DENEGACIÓN DE SERVICIO	4
ENUMERACIÓN	5
VIOLACIÓN DE ACCESO	6
VIOLACIÓN DE INTEGRIDAD	7
ENT. SISTEMA CORRUPTO	8
ENT. USUARIO CORRUPTO	9
ACTIVO EN PELIGRO	10
MODO DE USO SOSPECHOSO	11
VIOLACIÓN DE CONEXIÓN	12
BINARIO INESTABLE	13
INICIO DE SESIÓN	14

Tabla 3.2: Etiquetado para la tabla 3.1

La motivación de la propuesta consistía en la elaboración de un marco matemático unificado, basado en un único parámetro, con el que llevar a cabo la fusión de las alertas emitidas por varios IDS. De esta manera, el sistema podía separar las falsas alertas de las verdaderas. El parámetro elegido fue la probabilidad de que una alerta haya sido ocasionada por el mismo evento que las otras, permitiendo de esta manera su agrupamiento. La tabla 3.1 muestra el grado de similitud entre los diferentes tipos de alertas obtenido al integrar su sistema de correlación con el conocido IDS, EMERALD. La tabla 3.2 indica el significado de cada una de las etiquetas de la tabla 3.1.

Esto causó gran impacto en el área de la detección de intrusiones, y en el año 2002 A. Valdes publicó junto con D. Andersson et al. [5] una comparativa de los resultados obtenidos sobre un mismo escenario de pruebas, entre la capacidad de fusión de alertas de su sistema de correlación de alertas, en combinación con diversos IDS, destacando entre ellos EMERALD y Snort.

3.2.3. M-Correlator

Otra interesante propuesta fue presentada en el año 2002 por P. Porras et al. [6]. En ella se planteaba la necesidad de aplicar mecanismos de fusión a las alertas emitidas por los sistemas de seguridad heterogéneos distribuidos espacialmente (INFOSEC), como cortafuegos, IDS o herramientas de autenticación. Para ello se

elaboró el sistema de correlación M-Correlator, el cual requiere de la aplicación de una base del conocimiento sobre las amenazas, y una base del conocimiento sobre los sistemas protegidos.

La figura 3.2 muestra un resumen del algoritmo mediante el que se procesan las alertas. Al emitirse dichas alertas, en primer lugar atraviesan una etapa de filtrado de la información irrelevante con el fin de simplificar el procesamiento. Se busca alertas similares y se procede a su priorización en función de la base del conocimiento sobre amenazas y la base del conocimiento sobre sistemas protegidos. A continuación se clasifican, y por último se lleva a cabo su agregación.

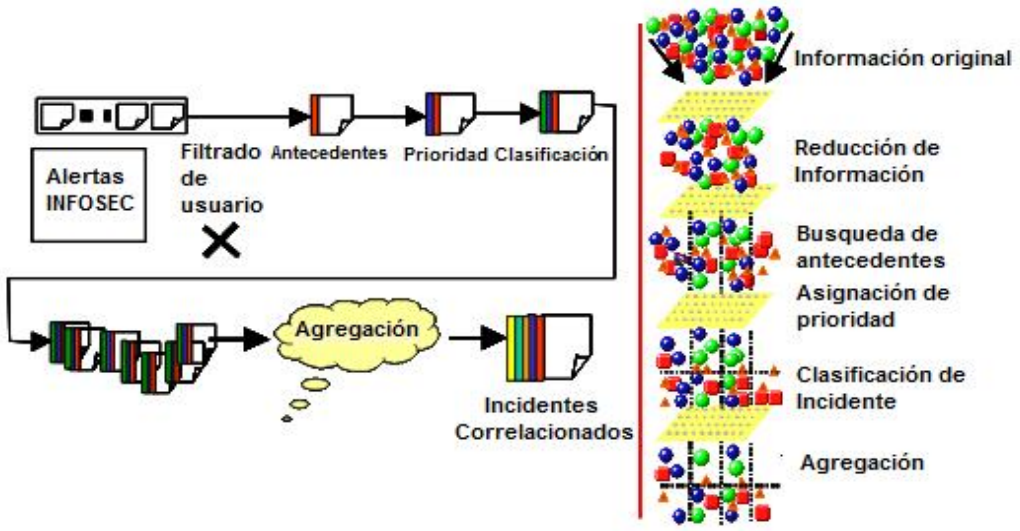

Figura 3.2: Esquema de M-Correlator

3.2.4. Correlación sobre BIDS

A la vista del auge en el desarrollo de este tipo de sistemas, el MIT Lincoln Laboratory también publicó su propia propuesta. O. M. Dain y R. K. Cunningham fueron los encargados de presentar un sistema de correlación probabilista [7] inspirado en el marco matemático planteado por A. Valdes y K. Skinner. Su motivación partía de la necesidad de optimizar el proceso de reconstrucción de los vectores que representaban los escenarios de ataques de la primera propuesta, tratando de llevar a cabo su elaboración de una manera similar a la que lo haría un operador experto y humano. Las tablas 3.3 y 3.4 muestran un ejemplo de cómo se reconstruye dicho vector. Partiendo de la suposición de que A,B,C,D y E representan las alertas

emitidas por los sensores, de una forma atómica, se rellenan diferentes tablas correspondientes a escenarios con diferente número de alertas. La tabla 3.3 contiene el ejemplo de reconstrucción de ataques de dimensión 4, y la tabla 3.4, del dimensión 5.

AE	ABE	ABCE	ACE
BE	BCE	CE	E

Tabla 3.3: Ejemplo de construccón de vector de ataque de dimensión 4

AE	ABE	ABCE	ACE
BE	BCE	CE	E
ADE	ABDE	ABCDE	ACDE
BDE	BCDE	CDE	DE

Tabla 3.4: Ejemplo de construccón de vector de ataque de dimensión 5

El error obtenido en los resultados era del 11,19 %, lo que para entonces significaba un funcionamiento muy deseable. Además, el despliegue sobre un escenario distinto de EMERALD, en este caso, el IDS conocido como BIDS (*Battlefield Intrusion Detection System*) desarrollado por su propio departamento, resultaba altamente innovador. A pesar de esto, la generación de tablas intermedias suponía una importante penalización al rendimiento del sistema que lo vuelve inefectivo para el procesamiento en redes de banda ancha.

3.2.5. La optimización del rendimiento de Ning y Xu

Un año después de la propuesta del el MIT Lincoln Laboratory, P. Ning y D. Xu resaltaron la necesidad de optimizar el comportamiento en tiempo real, tanto de los IDS, como de los sistemas de correlación de alertas. Para ellos se centraron en el segundo punto, y en las carencias de los marcos matemáticos publicados frente a ataques de complejidad, así como al problema de la reducción de la calidad de servicio del sistema protegido.

Su propuesta más destacada [8] se centró en la mejora del rendimiento de este tipo de sistemas, mediante la aplicación de estrategias de optimización comúnmente empleadas en el ámbito de las bases de datos. Dichas estrategias combinan el uso de estructuras de datos que reducen el impacto de las estructuras convencionales (*T Trees, Linear Hashing ..*) con métodos propios de consultas sobre bases de datos

(*nested loop join, sort join* ..). En su trabajo también se compara los experimentos obtenidos aplicando cada una de ellas, resaltando los beneficios e inconvenientes de implementar cada una de ellas en los sistemas de seguridad. Por último indican la necesidad de seguir avanzando en la investigación acerca de la mejora del rendimiento de este tipo de sistemas, y plantean algunas estrategias como posibles trabajos futuros.

3.2.6. LAMBDA Y CRIM

En el año 2000, F. Cuppens y R. Ortalo presentaron un lenguaje para modelar las bases de datos involucradas en el proceso de detección de alertas denomindo LAMBDA (*A Lenguage to Model a Databases for Detection of Attacks*) [58]. Dicho lenguaje se basaba en la representación lógica de las características de las intrusiones y en la representación mediante operaciones algebraicas de cada uno de los pasos del proceso de intrusión. De esta manera, de la misma forma que para entonces el CISL (predecesor del estándar IDMEF) unificaba el formato de alertas emitido por los distintos sensores, LAMBDA pretendía unificar el formato en el que las bases de datos almacenaban la información de las amenazas necesaria para llevar a cabo una detección por firmas. La figura 3.3 muestra un ejemplo de ataque MIR-0163 modelado con el lenguaje LAMBDA.

```
<?xml version="1.0" encoding="UTF-8"?>
<attack attackid="MIR-0163">
<name>mount partition</name>
<pre>access_level(Source_user,Target_address,remote),
      mounted_partition(Target_address,Partition),
</pre>
<post>can_access(Source_user,Partition)
</post>
<scenario>Action</scenario>
<cond_scenario>
 script(Action,'mount -t nfs $Partition:$Target_address $Partition')
</cond_scenario>
<detection>Alert</detection>
<cond_detection>alert(Alert),
                source(Alert,Source),
                source_user(Source,Source_user),
                target(Alert,Target),
                target_node(Target,Target_node),
                address(Node,Target_address),
                classification(Alert,"MIR-0163")
</cond_detection>
</attack>
```

Figura 3.3: MIR-0163 modelado con el lenguaje lambda

En base a este lenguaje, F. Cuppens presentó el sistema de correlación de alertas CRIM [10], involucrado en el proyecto MIRADOR. Dicho proyecto fue soportado por la Agencia de Defensa Francesa DGA (*Direction Générale de lArmement*) en colaboración con la empresa Alcatel y 3 importantes laboratorios de investigación: ONERA, ENST-Bretagne and Supelec. CRIM consistía en un módulo suplementario los IDS, que correlacionaba las alertas en función de una base del conocimiento modelada con el lenguaje LAMBDA. En la figura 3.4 se indica la arquitectura de CRIM.

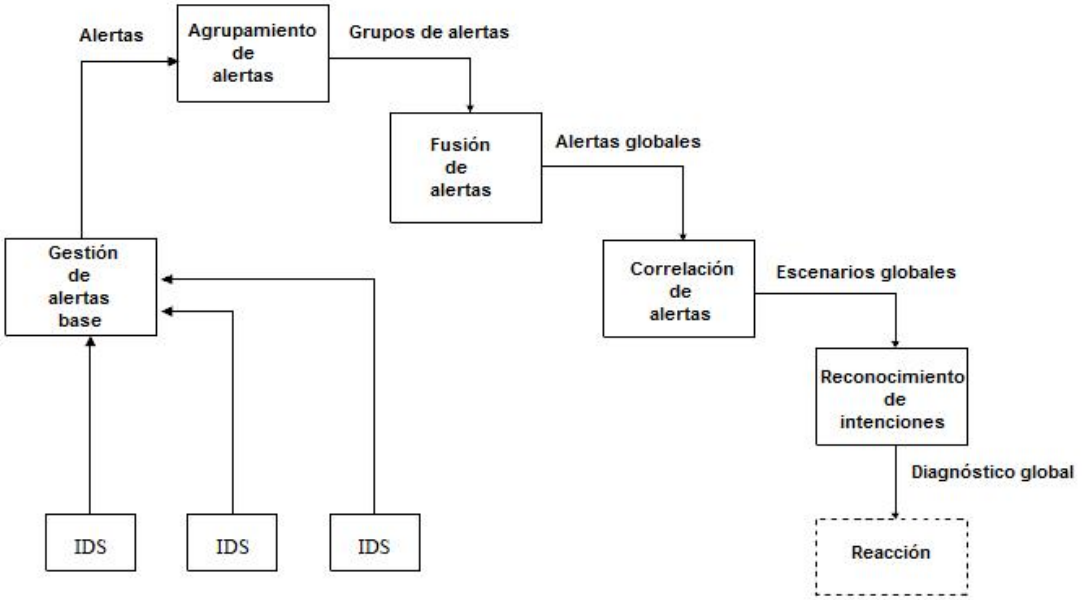

Figura 3.4: Arquitectura de CRIM

En CRIM las alertas emitidas por los sensores son transformadas al lenguaje IDMEF en el módulo de gestión de alertas base. Para reducir el procesamiento de la información son agrupadas en función de criterios marcados por la base del conocimiento sobre ataques, y se fusionan. A continuación se lleva a cabo la correlación, aplicando la base de datos de ataques modelada con LAMBDA, y en función de los resultados y las intenciones de la intrusión, se toman las medidas preventivas necesarias para mitigar la amenaza.

Tanto el lenguaje LAMBDA como la propuesta MIRADOR tuvieron gran impacto en el área de la investigación de intrusiones. No obstante, el propio Cuppens se daría cuenta de que la detección basada en firmas resultaba insuficiente para hacer frente a la rápida proliferación de las amenazas, y desvió su línea de investigación a la detección y correlación de intrusiones basadas en anomalías.

3.2.7. Modelado en función del objetivo y el proceso de intrusión

F. Cuppens et al [9] llevaron a cabo una nueva propuesta en el área de la correlación de intrusiones, basándose en sus experiencias con el sistema CRIM, y en el lenguaje de modelado de ataques LAMBDA. Para ello enfocaron los criterios de correlación, en las precondiciones marcadas por las necesidades del proceso de in-

trusión, y en las postcondiciones que representan las consecuencias del ataque.

Adicionalmente, y dado el impacto de los falsos positivos en sus primeras propuestas, se propone un mecanismo para su mitigación. Se basa en una ligera modificación del lenguaje LAMBDA para llevar a cabo lo que denominaron: correlación abductiva. La correlación abductiva permite al sistema generar falsas alertas con el fin de completar los escenarios de las alertas que no han podido ser asociadas a ninguno de ellos. De esta manera resulta mucho más sencillo identificar qué alertas son causadas por situaciones desconocidas (en este caso, falsos positivos) y qué alertas corresponden a verdaderas amenazas.

3.2.8. Correlación basada en la reconstrucción de escenarios mediante grafos de distancias

En el 2004, Noel et al. publicaron un trabajo referente [12] en el campo de la correlación de alertas basada en la reconstrucción de escenarios de ataques. En él se propone por primera vez el uso de grafos de distancias para la representación de los escenarios. La mayor ventaja que aportaba su uso era la importante mejora en la eficiencia del procesamiento de escenarios, debido a la reducción del número de operaciones de cómputo a un reducido subconjunto de cálculos aritméticos. De esta manera se dejó de lado el uso de reglas lógicas con dicho propósito.

La figura 3.5 muestra la derivación de un grafo de ataque a un grafo de eventos. El grado del ataque viene dado por las precondiciones de la amenaza, y corresponde al modo de vista del atacante, ya que incluye los *exploits* que tendrá que aplicar para completar el ataque con éxito. El grafo de eventos corresponde al modo de vista del defensor, e identifica los distintos eventos que generarán las alertas. Es muy importante tener en cuenta las distancias entre eventos, ya que cuando son cortas, se trata de una amenaza más probable. Esto quiere decir que se requerirá un número menor de pasos intermedios para llegar a la siguiente etapa de la intrusión.

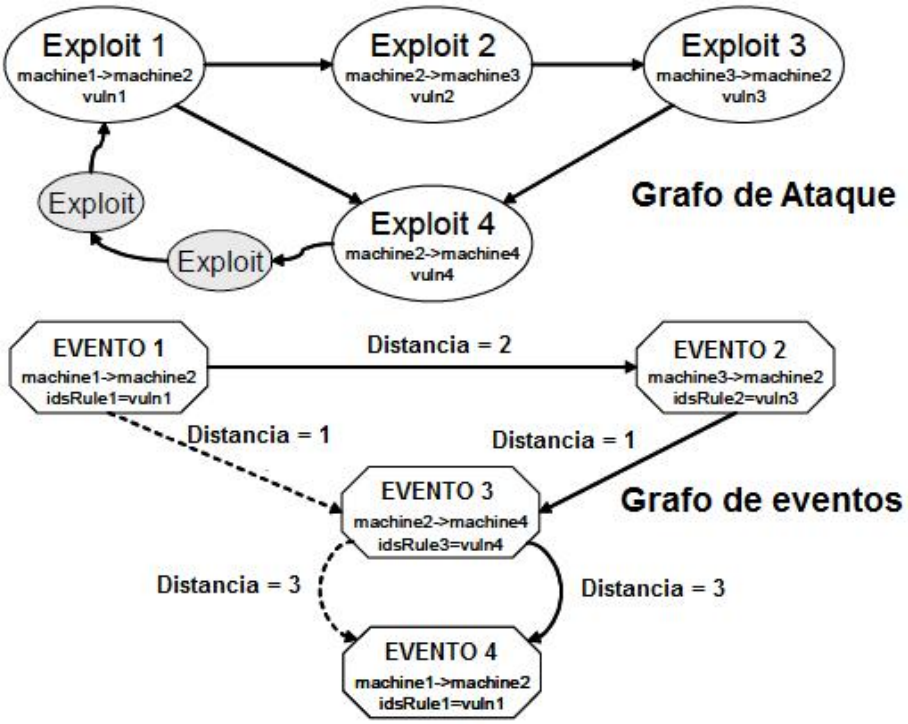

Figura 3.5: Ejemplo de generación de un grafo de eventos

Los experimentos realizados demostraron que el rendimiento era mucho mejor, procesando la amenaza más larga en 24 milisegundos. Además, la sobrecarga del sistema apenas llegó al 30 % en los peores casos. Al llegar la decadencia de la detección basada en firmas, esta propuesta tuvo mucho éxito en otra área de la seguridad: la del análisis de riesgos.

3.2.9. Reducción del número de reglas basándose en la extracción de la estrategia de ataque

En el año 2006, B. Zhu, A. A. Ghorbani [13] criticaron el planteamiento de la mayor parte de los trabajos de correlación de alertas presentados hasta el momento. Afirmaban que se centraban en agrupar e interrelacionar las alertas emitidas por los sensores desde el punto de vista del defensor, pero que pasaban por alto la estrategia del atacante en sí.

En consecuencia, proponen una estrategia de extracción de dichas estrategias sin tener ningún conocimiento previo sobre ellas. De esta manera, se dejaban de lado las enormes bases de firmas cuyo tratamiento generaba un enorme esfuerzo computacional, y se establecía una alternativa eficiente, acorde con la reciente tendencia a

la detección basada en anomalías.

El sistema publicado proponía diferentes implementaciones basadas en redes neuronales MLP (*Multilayer Perceptron*) y máquinas de vector soporte SVM. La salida probabilista de estas herramientas permite determinar cuáles de las alertas tratadas deben de ser correlacionadas. Además incorporaba una estructura denominada matriz de correlación de alertas ACM (*Alert Correlation Matrix*), mediante la que se representaba el grado de correlación entre cada par de alertas. Dichas matrices eran rellenadas en una etapa de entrenamiento, y permitían determinar con qué alertas correlacionar los eventos auditados. A partir de los datos obtenidos era posible construir el grafo del ataque.

	a1	a2	a3	a4	a5
a1	*C(a1,a1)*	*C(a1,a2)*	*C(a1,a3)*	*C(a1,a4)*	*C(a1,a5)*
a2	*C(a2,a1)*	*C(a2,a2)*	*C(a2,a3)*	*C(a2,a4)*	*C(a2,a5)*
a3	*C(a3,a1)*	*C(a3,a2)*	*C(a3,a3)*	*C(a3,a4)*	*C(a3,a5)*
a4	*C(a4,a1)*	*C(a4,a2)*	*C(a4,a3)*	*C(a4,a4)*	*C(a4,a5)*
a5	*C(a5,a1)*	*C(a5,a2)*	*C(a5,a3)*	*C(a5,a4)*	*C(a5,a5)*

Figura 3.6: Ejemplo de matriz ACM para 5 ataques

La figura 3.6 muestra un ejemplo de matriz de correlación ACM para 5 alertas $(a_1, a_2, a_3, a_4, a_5)$, en la que $C(a_i, a_j)$ representa la relación temporal entre las alertas a_i y a_j. En los experimentos que realizaron con las trazas de tráfico DARPA 2000 se arrojaron unos excelentes resultados, sembrando de esta manera las bases para futuros trabajos próximos al reconocimiento de anomalías.

3.2.10. Correlación de alertas en tiempo real contra ataques de denegación de servicio

R. Sadoddin y A. Ghorbani [14] presentaron en el 2009 una propuesta para la correlación de alertas en tiempo real. Su principal motivación fue el establecimiento de un marco para la gestión de alertas online, basado en la estrategia FSM (*Frequent Structure Mining*), y centrado en la detección de ataques de denegación de servicio distribuidos.

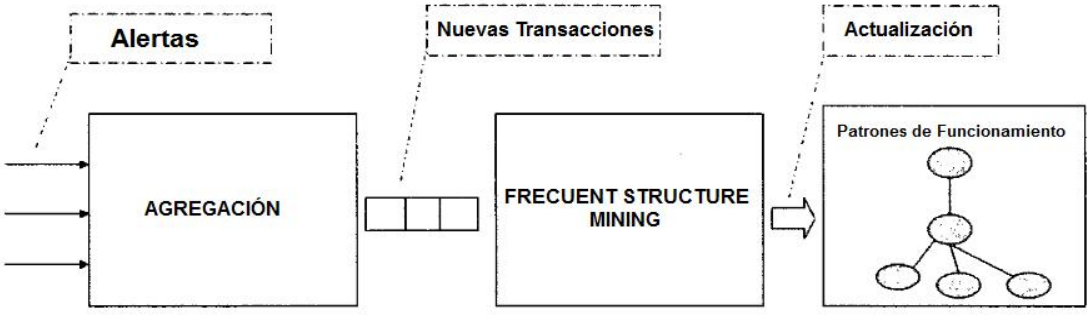

Figura 3.7: Esquema para la correlación incremental

En la figura 3.7 se muestra su propuesta de correlación incremental. En una primera etapa se generan patrones en forma de grafos, que contienen información acerca del origen y el objetivo de la alerta. La segunda etapa se encarga de considerar en tiempo real, la frecuencia y el momento de aparición de cada una de ellas. Además se actualiza continuamente acorde a nuevas detecciones. Por último se generan árboles que representan los patrones de comportamiento del proceso de intrusión.

Este proceso les permite obtener una precisión del 96 % en el análisis de las trazas de tráfico DARPA 2000, y un comportamiento en tiempo real. La propuesta resulta pionera en la adaptación de las estrategias de correlación contra la denegación de servicio distribuida, y plantea un marco que va a ser tenido en consideración en futuros trabajos.

3.2.11. Tendencias actuales

Una de las dos tendencias actuales en el campo de la correlación de alertas deriva de la necesidad de adaptar los sistemas de defensa, a las redes de banda ancha y fibra óptica. La aparición de redes cada vez más rápidas requieren de sistemas

defensivos eficientes, que ocasionen un impacto bajo sobre su calidad de servicio. El uso de grandes bases de datos sobre escenarios de ataques queda obsoleto, y la tendencia lleva a la implementación de distintas estrategias de minería de datos. Estas estrategias normalmente se basan en técnicas de aprendizaje automático, capaces de correlacionar las alertas emitidas por las anomalías en tiempo real.

Una de las estrategias más elegidas es la lógica difusa. Dada la naturaleza de este tipo de sistemas, resulta intuitivo plantear una correlación en base a funciones de pertenencia a grupos difusos. En esta rama cabe destacar el trabajo de H. T Elshoush et al.[15] para la reducción de la tasa de falsos positivos del IDS, o el de K. Alsubhi et al.[16] para la priorización del tratamiento de alertas.

El trabajo de R. Sadoddin y A. Ghorbani influyó sobre nuevos trabajos basados en el uso de redes neuronales artificiales, como el de G. Tjhai et al. [17]. Esta propuesta combina el uso de mapas auto-organizativos SOM con el algoritmo de agrupamiento K-means, para determinar si una alerta era verdadera o falsa.

S. Peddabachigaria et al. [18] propusieron un sistema de correlación de alertas que combinaba máquinas de vector soporte SVM con árboles de decisión. De esta manera se alcanzaba una excelente precisión en un periodo corto de procesamiento. Motivados por este trabajo, H. Nguyen et al. [19] publicaron un estudio entre el uso de diferentes estrategias de minería de datos sobre los dos niveles propuestos.

Cabe resaltar el uso de otras herramientas, como los algoritmos genéticos [20] y o los agentes [21], con el fin de considerar dichos propósitos.

La otra tendencia actual en este área son las propuestas referentes a la correlación de alertas en Sistemas de Detección de Intrusiones distribuidos, orientados a brindar protección a sistemas desplegados mediante arquitecturas de computación *Cloud* y *Ad Hoc* [22] [23] [24].

3.3. Clasificación de los sistemas de correlación de alertas en base a su comportamiento

Las estrategias de correlación de alertas se dividen en cinco categorías muy diferentes [59], tomando como criterio de clasificación los objetivos y el comportamiento de cada una de ellas. Su relevancia ha llevado a los investigadores a aprovechar el

enfoque propuesto por los primeros trabajos, innovando mediante el uso de estrategias avanzadas de minería de datos, *soft-computing* o sistemas multi-agente. Esta sección se centra en explicar las principales características de cada una de ellas y los trabajos pioneros que han dado pie a su consolidación.

La figura 3.8 muestra el esquema de la clasificación de los sistemas de correlación de alertas, basada en las cinco categorías que se explican a continuación.

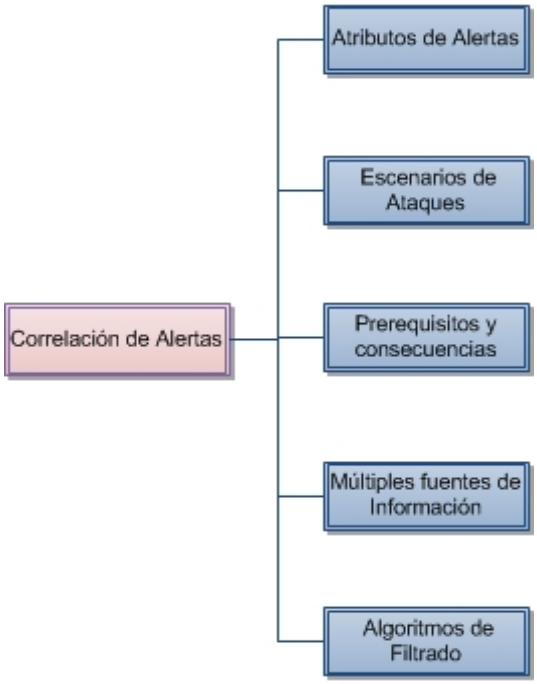

Figura 3.8: Clasificación de los sistemas de correlación de alertas

3.3.1. Aproximaciones basadas en la similitud entre los atributos de las alertas

Las aproximaciones contenidas en esta categoría basan su funcionamiento en el estudio de la información que proveen los atributos de las propias alertas. Cada tipo de sensor provee diferente información. Por ejemplo, un NIDS indica las direcciones IP, el puerto o el protocolo del tráfico que ha generado la alerta, mientras que un HIDS indica atributos como el proceso, los registros o las llamadas al sistema involucradas en el proceso de intrusión. Todos los trabajos incluidos en esta categoría también tienen en común la tendencia al agrupamiento de las alertas mediante diferentes estrategias y funciones para demarcar la distancia que separa unas agrupaciones de otras. Los dos trabajos más importantes pertenecientes a esta categoría, son el de Valdes y Skinner [3], y el de F. Cuppens [60]. Estas propuestas marcaron las pautas a seguir en trabajos futuros basados en la similitud entre los atributos de las alertas. A continuación se explica brevemente cada una de estas dos tendencias:

Correlación de alertas probabilista

Esta aproximación deriva del trabajo de Valdes y Skinner [3]. Se centra en establecer la similitud entre las alertas emitidas por el sensor en base a los atributos

que se superponen, es decir, los que son iguales. Las diferencias se etiquetan en una escala de 0 a 1, en el que 0 indica una diferencia total, y 1 indica que son exactamente la misma alerta. Tras llevar a cabo varias pruebas con diversas funciones de similitud, los resultados arrojados fueron especialmente buenos, llegando a agrupar 4439 alertas en 604 alertas. El conjunto de datos de los experimentos acabó por convertirse en un estándar funcional.

Agrupación y fusión de alertas de múltiples sensores

F. Cuppens [60] fue el pionero en esta aproximación, proponiendo una estrategia de correlación de alertas que involucraba diferentes sensores, enmarcada en el proyecto MIRADOR. Para normalizar su contenido, partió de la premisa de que las alertas pasaban por una etapa de normalización al estándar IDEMF [1] para el intercambio de alertas. La arquitectura del sistema planteado constaba de un módulo para gestionar todas las alertas emitidas, un módulo para establecer el agrupamiento, y un módulo para llevar a cabo su fusión. Los resultados obtenidos mediante la combinación de alertas de dos IDS (Snort y e-Trust) resultaron especialmente favorables, llegando a agrupar más de 300 alertas en casi 100 agrupaciones.

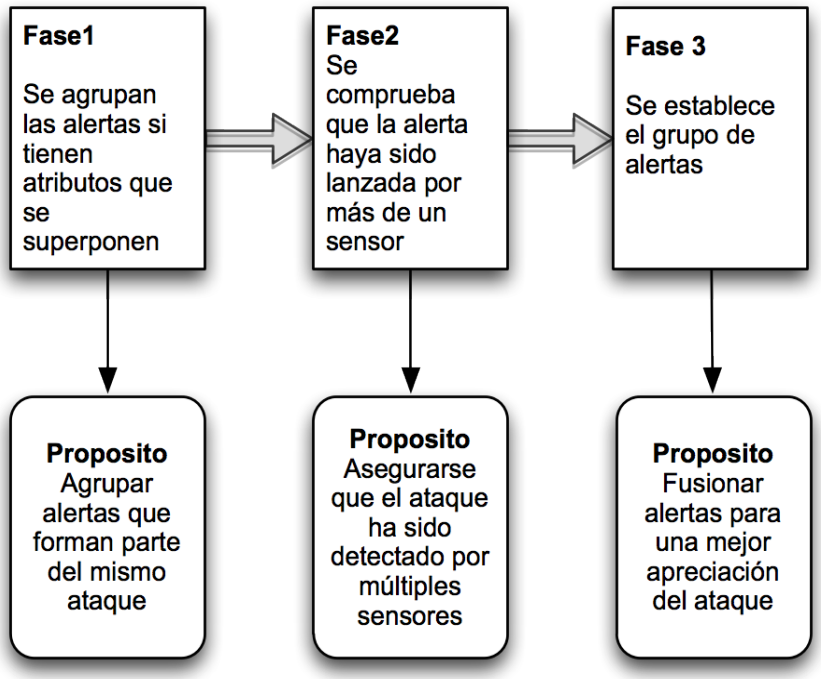

Figura 3.9: Esquema de fusión de alertas

La figura 3.9 muestra las etapas para la fusión de alertas más comunes.

3.3.2. Aproximaciones basadas en escenarios de ataque predefinidos

La agrupación de las alertas en función de escenarios de ataques predefinidos también ha dado pie a diversos trabajos. En algunos de ellos han sido especificados por los usuarios, mientras que en otros han sido generados mediante procesos de aprendizaje automático. Aunque esta estrategia ha resultado especialmente eficaz contra ataques conocidos, existe bastante polémica acerca de su precisión contra ataques desconocidos. Además implica la necesidad de bases de datos con los diferentes escenarios que en algunos casos, pueden incrementar el coste de construcción del sistema [61]. Los trabajos más importantes de esta categoría son el de Debar y Wespi [2] y el de Morin y Debar [62], el primero de ellos basado en la agregación y correlación de las alertas en sistemas IBM/Tivoli, y el otro en los formalismos de Chronicles.

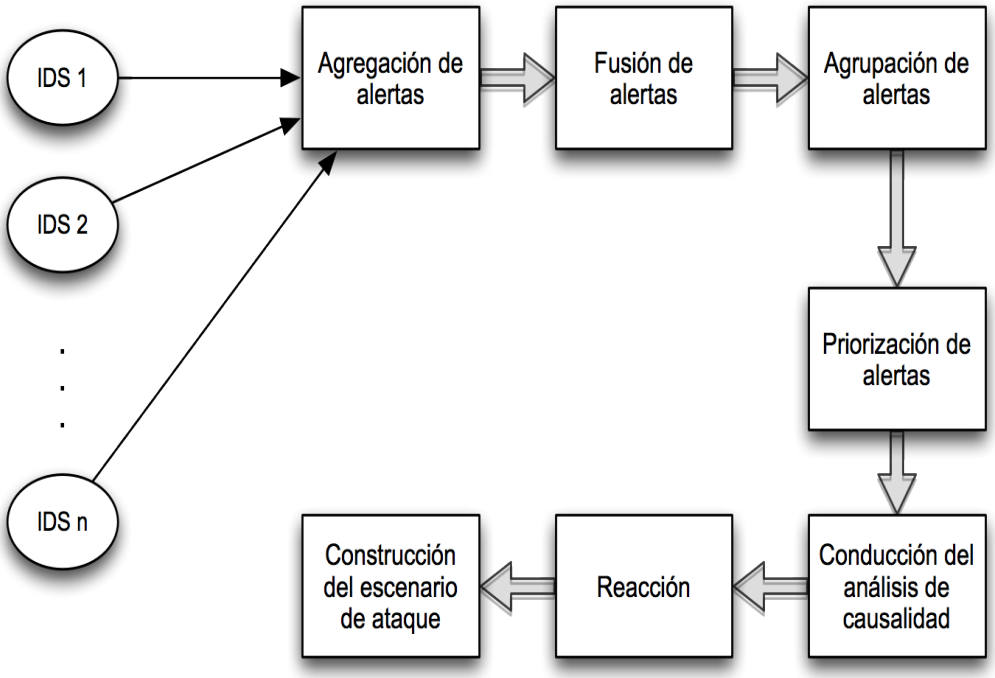

Figura 3.10: Esquema para la correlación basada en escenarios predefinidos

La figura 3.10 muestra un esquema básico para la correlación mediante la construcción de escenarios de ataques.

Agregación y correlación basada en escenarios

El trabajo de Debar y Wespi [2] propuso diferentes pautas de diseño en el ámbito de la agregación y correlación basada en escenarios. Entre ellas destacan su propuesta de arquitectura para la combinación de múltiples IDS, el modelado de las alertas emitidas, y los componentes de agregación y correlación ACC (*Aggregation and Correlation Components*). La etapa ACC estaba dividida en tres fases: una de preprocesamiento de las alertas para normalizar su formato, una de correlación con la que se establece si un incidente estaba relacionado con otro previo, describiendo así el escenario del ataque, y una última etapa de agregación de alertas en base a los escenarios creados.

Correlación basada en el modelo Chronicles

El trabajo de Morin y Debar [61] fue pionero en la correlación de alertas a través de sistemas Chronicles. Este tipo de sistemas permite el modelado de patrones de eventos temporales en sistemas dinámicos, pudiendo de esta manera auditar su evolución. Su incorporación al área de la detección de intrusiones tuvo como objetivo mitigar los ataques de denegación de servicio basados en desbordamientos de alertas, etiquetando de una manera efectiva las verdaderas amenazas y los falsos positivos.

3.3.3. Aproximaciones basadas en los prerrequisitos y las consecuencias de los ataques

Las aproximaciones basadas en los prerrequisitos y las consecuencias de los ataques, normalmente son de características multi-etapa y tienen como objetivo solucionar el problema de la correlación de alertas cuando las amenazas son desconocidas. Este tipo de alertas son típicamente emitidas por IDS basados en anomalías, y mediante esta estrategia se trata de reconstruir escenarios de cualquier complejidad, mediante la identificación de las diferentes etapas del ataque. A diferencia de la aproximación basada en escenarios predefinidos, en este caso se desconoce su naturaleza, y por lo tanto, se van construyendo sobre la marcha. Dicha construcción es llevada a cabo mediante la aplicación de sencillas expresiones lógicas de primer orden, o mediante lenguajes específicos para el modelado de ataques, como LAMBDA [10].

Las diferentes propuestas han resultado muy precisas frente a amenazas desconocidas, pero tienen el inconveniente de que en muchos trabajos se pasa por alto las alertas que no pueden ser correlacionadas. Además son muy sensibles ante la inyección de falsos positivos por parte del atacante con el fin de forzar la construc-

ción de escenarios incorrectos. Esta situación se ha de prevenir a la hora de llevar a cabo un diseño eficaz. La figura 3.11 muestra un esquema básico para la correlación considerando las consecuencias de las amenazas. A continuación se explican algunos de los trabajos y tendencias más representativos

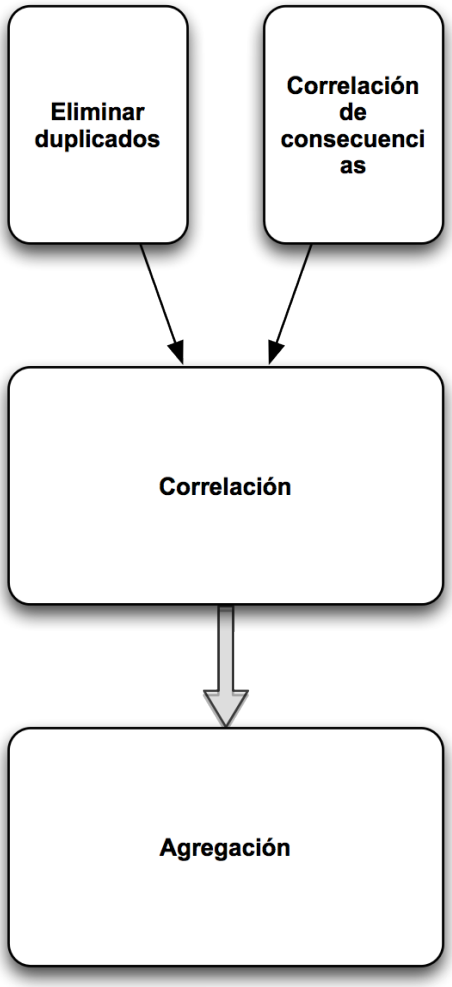

Figura 3.11: Esquema para la consideración de las consecuencias de las amenazas

Aproximaciones basadas en precondiciones y postcondiciones

El trabajo de Cupens y Miege [10] propuso un sistema de correlación para las alertas emitidas por diferentes IDS. La importancia de su propuesta para esta categoría, reside en ser pionera en la construcción de escenarios basándose en precondiciones y postcondiciones de los ataques, y en su modelado mediante el lenguaje LAMBDA. A partir de él se modelan expresiones que van a permitir evaluar si el ataque puede llegar a completarse con éxito, así como sus consecuencias. Además de esto, se

propone una estrategia de mitigación de las tasas altas de falsos positivos utilizando lo que denominaron correlación abductiva, que consiste en la generación de alertas virtuales que faciliten la correlación de los falsos positivos y su etiquetado.

Aproximaciones basadas en prerrequisitos y consecuencias

Ning et al. [63] establecieron una nueva aproximación basada en los prerrequisitos y consecuencias de los ataques, que posteriormente fue extendida en posteriores trabajos. Para ello establecían el tipo de ataque mediante una tripleta que contenía información sobre el ataque, una fórmula lógica que establece los prerrequisitos del ataque, y las consecuencias a las que se llegan a partir de un conjunto de predicados. De esta manera, después de la derivación de los prerrequisitos y consecuencias lógicas de cada ataque, se establecen asociaciones mediante grafos de correlación que facilitan su etiquetado, y permiten llevar a cabo la agrupación de alertas similares.

Aproximaciones basadas en hipótesis de ataques y técnicas de razonamiento

Esta nueva aproximación [64] surge de una propuesta que originariamente extiende la aproximación basada en prerrequisitos y consecuencias de Ning et al. [63] previamente mencionada, orientada a la reducción de la tasa de falsos positivos del IDS. Apareció a partir de la observación de que cuando ciertos ataques intermedios son detectados por el IDS, el escenario de ataque que se está construyendo tiende a dividirse en múltiples escenarios diferentes. De esta manera, plantean que si las alertas de dichos escenarios múltiples satisfacen ciertas condiciones de igualdad, podrían pasar a ser hipótesis, y por lo tanto ser derivadas a partir de sus atributos.

Las hipótesis podrán ser validadas o no en base a sus atributos y el contexto en que se desencadenaron, consolidándose únicamente aquellas hipótesis validadas. Los resultados de sus experimentos demostraron que descartar las hipótesis no validadas permite reducir con eficiencia la tasa de falsos positivos. A pesar de los buenos resultados, ha habido mucha controversia en torno al uso de dicha aproximación. Aplicar esta técnica penaliza el rendimiento del sistema, situación no deseable si se pretende llevar a cabo un etiquetado en tiempo real. Una vez más, el atacante puede aprovecharse de ello para inyectar falsas alertas en el sistema y denegar su servicio.

3.3.4. Aproximaciones basadas en fuentes de información múltiples

Para proteger un sistema, normalmente se despliegan diferentes mecanismos de defensa a nivel de red y a nivel de *host*, a lo largo de distintos perímetros de seguridad. La gran ventaja de esta asociación reside en que determinados mecanismos defensivos resultas más eficaces frente a determinadas amenazas que otros, complementándose de esta manera su eficacia. Esta aproximación se centra en la correlación de alertas de aquellos sensores de características muy diferentes, como pueden ser un NIDS y un HIDS, y en su complementación. Los trabajos más representativos de este área son el de P. Porras et al. [6] basado en la misión y el impacto de los intentos de intrusión, el modelo M2D2 de Morin et al. [65] y la aproximación basada en la correlación de eventos en función de los recursos comunes de Xu. y Ning [66]. A continuación se explica brevemente cada uno de ellos.

Aproximación basada en el objetivo y el impacto de los ataques

El trabajo de P. Porras et al. [6] propuso por primera vez la correlación de las alertas generadas por múltiples sensores en base al objetivo de las amenazas y su impacto, mediante el sistema M-Correlation, anteriormente mencionado. Para llevar a cabo la correlación emplearon dos bases del conocimiento. La primera de ellas contiene toda la información referente a los ataques y vulnerabilidades a los que se expone el sistema, y la segunda, incluye la información acerca del sistema protegido y la topología de la red en que se ubica. De esta manera, cuando alguno de los sensores emite una alerta, se procede en primer lugar al filtrado de las alertas de mayor y menor impacto sobre el sistema protegido. Para ello se otorga una puntuación a cada una de ellas, en función de dichas bases del conocimiento. Esta puntuación va a permitir la definición de la prioridad de tratamiento de las alertas y la del peligro potencial que representan para el sistema. En base a estos dos criterios, se llevará a cabo el agrupamiento de alertas.

El modelo M2D2

Morin et al. [65] propusieron el modelo M2D2, a partir del cual se normalizaban diferentes características de las alertas generadas por los sensores, como por ejemplo el contexto en el que se ubica el sistema protegido, las herramientas de seguridad disponibles, el grado de protección de la información estimado, y por supuesto las alertas emitidas. En base a este modelo, se llevarían a cabo las acciones de agrupamiento y correlación. Este tipo de modelado fue la base del estándar IDEMF [1]

para la normalización de alertas.

Aproximación a la correlación de eventos en función de los recursos comunes

La propuesta de Xu y Ning [66] para la agrupación de eventos en función de los recursos de entrada y salida, se lleva a cabo en tres etapas. En primer lugar se agrupan las alertas generadas por eventos similares. En segundo lugar se considera si los grupos de alertas generados en el paso previo son consistentes con el sistema protegido, suponiendo en tal caso una amenaza. Por último, para aquellos grupos que resulten peligrosos se construye el escenario del ataque en función de los recursos de entrada y salida necesarios para su éxito, permitiendo así una nueva agrupación aún más precisa. Los resultados arrojados por esta aproximación fueron especialmente buenos, y a partir de ella aumentó la cantidad de propuestas basadas en diferentes agrupamientos en distintos niveles, combinando diferentes aproximaciones.

3.3.5. Aproximaciones basadas en algoritmos de filtrado

La idea de filtrar las alertas emitidas por los sensores tiene como finalidad reducir el impacto en el rendimiento del sistema de correlación de alertas, y reducir la tasa de falsos positivos encadenada por la estimación de alertas de baja relevancia. La propuesta de Xu y Ningcite [66] ha sido un claro precedente a esta aproximación, a pesar de tener como principal objetivo el tratamiento de alertas generadas por múltiples sensores. En la actualidad esta es la tendencia predominante. El rápido incremento del ancho de banda de las redes y la capacidad de cómputo de los procesadores ha llevado a la necesidad de separar la información que va a ser tratada y la que no, con el fin de restar carga al sistema. El uso de herramientas como la lógica difusa [16], mapas auto-organizativos SOM [17] o máquinas de vector soporte SVM [18], son un ejemplo de la tendencia actual a la aplicación de técnicas de agrupamiento, minería de datos y *Soft-Computing* en el área de la correlación de alertas.

Capítulo 4

APACS: Sistema de correlación de alertas para NIDS basados en anomalías que analizan la carga útil del tráfico

La motivación de esta parte del trabajo ha sido llevar a cabo un diseño capaz de cubrir las necesidades de los NIDS basados en anomalías que analizan la carga útil del tráfico de la red. Los principales objetivos han sido la identificación de los falsos positivos, la valoración cualitativa de la probabilidad de que una anomalía sea efectivamente una amenaza, y la clasificación cuantitativa que muestre el tipo concreto de amenaza detectada. Esta última clasificación es llevada a cabo considerando ataques a nivel de paquete o a nivel de trazas de tráfico, siendo esta última opción una alternativa a la reconstrucción de escenarios de ataques.

Para ello se proponen pautas de diseño que pueden ayudar a futuros desarrolladores y que pueden ser aplicables en diferentes bases de conocimiento sobre ataques. De ahora en adelante, el sistema propuesto será denominado APACS (*Anomaly Payload Alert Correlation System*).

4.1. Aproximación general

Cuando un IDS analiza el tráfico de una red, normalmente genera una cantidad muy grande de falsos positivos. A pesar de que el NIDS haya sido diseñado para disminuir el porcentaje de dichas alertas, la cantidad de información que se genera sigue siendo muy alta, y por consiguiente, difícil de tratar. La repercusión del

número de falsos positivos tiene tal impacto, que incluso han aparecido técnicas de evasión de NIDS basadas en ello. Un ejemplo de esta aplicación, es el ataque denominado *Alert Flood*, traducido como desbordamiento de alertas. En el año 2001 apareció la primera herramienta conocida que trataba de evitar un IDS mediante desbordamiento de alertas, denominada Stick [67]. Su funcionamiento se basaba en saturar el canal de control que comunicaba el IDS con el operador por medio de una inyección de falsas alertas, denegando de esta manera su servicio. En el trabajo de G. Tedesco y U. Aickelin [68] se explican variantes de *Alert Flood* cuyo objetivo no se limita a denegar el servicio del IDS, sino que también permiten ofuscar la naturaleza del ataque verdadero en una maraña de falsos ataques. Estos mismos autores proponen una estrategia de correlación de alertas basada en los algoritmos conocidos como *Bucket Token* comúnmente empleados para el control del flujo de tráfico en los protocolos de red. El peligro de un ataque de desbordamiento de alertas y la dificultad de interpretar la información generada por el IDS en situaciones reales de tráfico, ha llevado a establecer la priorización de la alertas generadas por un NIDS como uno de los dos grandes objetivos a cubrir a la hora de llevar a cabo el diseño de uno de estos sistemas.

El otro gran objetivo a cubrir es la clasificación del contenido de las alertas acorde a una base del conocimiento. De esta manera, el operador no solamente conoce el grado de peligrosidad del paquete que ha generado la alerta, sino que también conoce la naturaleza de la misma. El primer criterio de clasificación es especialmente bueno para poder tomar decisiones rápidas de descarte de paquetes, mientras que el segundo, además va a permitir un mejor conocimiento de las estrategias llevadas a cabo por el atacante.

El diseño de APACS consiste en un modelo híbrido. APACS consta de un bloque encargado de establecer la probabilidad de que una alerta se trate de una verdadera amenaza, y de otro bloque, cuya función es establecer la naturaleza de la misma. El módulo encargado de establecer la probabilidad del ataque, utiliza técnicas de agrupamiento conocidas como *clustering*. Como se explica en la siguiente sección, las clasificaciones a las que da lugar APACS se fundamentan en el establecimiento de grupos de riesgo dependientes de la distancia entre los parámetros procedentes de las reglas generadas por el NIDS y los valores generados en la fase de detección. De esta manera se decide si la anomalía tiene pretensiones maliciosas o no. Por otro lado, el módulo encargado de establecer la naturaleza del ataque consta a su vez de dos fases que atienden dos posibles amenazas.

La primera fase se encarga de clasificar la anomalía a nivel de paquete en función de las reglas del conjunto de ataques utilizados para el entrenamiento. La clasificación se lleva a cabo mediante una sencilla red neuronal. En combinación con el bloque encargado de establecer la probabilidad del ataque, y gracias a las ventajas computacionales de las redes neuronales, es posible establecer una primera clasificación en tiempo real. Esta clasificación es esencial a la hora de tomar decisiones sobre qué hacer con el tráfico sospechoso que fluye a través del NIDS.

La segunda fase de este módulo tiene como objetivo establecer una clasificación a nivel de traza de tráfico. De esta manera es posible desenmascarar ataques distribuidos en distintos paquetes capaces de reconstruirse al llegar al sistema víctima, o de llevar a cabo un análisis forense más detallado de una posible amenaza. Esta última opción resulta especialmente interesante cuando el NIDS está integrado en una *Honeynet*. Su modelado es llevado a cabo mediante un sistema de encaje de patrones implementado con un algoritmo genético, decisión potenciada por la necesidad de disponer de características no deterministas, y la probabilidad de que la anomalía pertenezca a cada una de las posibles soluciones.

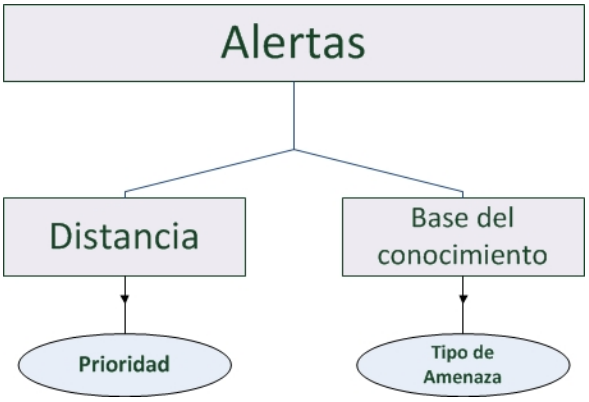

Figura 4.1: Esquema de clasificación a nivel de paquete

La figura 4.1 muestra el esquema de las fases en que se lleva a cabo la clasificación a nivel de paquete. El módulo que etiqueta las probabilidades se basa en estrategias de agrupamiento. El tipo del ataque es establecido mediante una red neuronal convencional, siendo esta la primera fase del módulo encargado de establecer la clasificación de la amenaza.

Por su parte, la figura 4.2 muestra el esquema de las fases en que se lleva a cabo

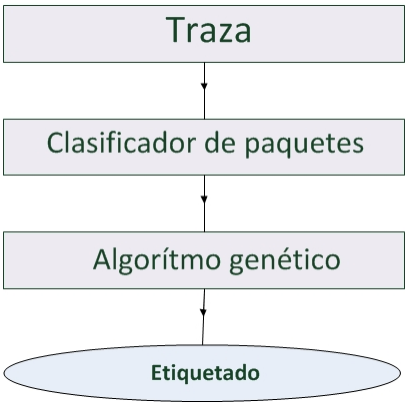

Figura 4.2: Esquema de clasificación a nivel de traza

la clasificación a nivel de traza. El algoritmo genético es la segunda fase de la clasificación del tipo del ataque, y tiene de entrada las salidas de las clasificaciones a nivel de paquetes para cada una de las traza a analizar. Una vez generadas las alertas y establecidas sus clasificaciones, el sistema emite la salida conforme al formato estandarizado de intercambio de mensajes entre Sistemas de Detección de Intrusiones conocido como IDMEF [1]. De esta manera se facilita su integración como sensor en sistemas *Honeynet* y se permite su interoperabilidad con otros sistemas de correlación de alertas, como por ejemplo el conocido Acarm-ng [69], que además puede aportar su interfaz gráfica.

```
<?xml version="1.0" encoding="UTF-8"?>
<idmef:IDMEF-Message version="1.0" xmlns:idmef="http://iana.org/idmef">
<idmef:Alert messageid= "43259">
<idmef:Analyzer analyzerid= "APAP">
<idmef:Node category= "tcp">
<idmef:location>APAP host</idmef:location>
<idmef:name>www.alkahest.es</idmef:name>
</idmef:Node>
</idmef:Analyzer>
<idmef:Source ident="69.5.88.225">
<idmef:Node ident="69.5.88.225">
<idmef:Address ident="69.5.88.225" category="ipv4-addr">
```

Figura 4.3: Ejemplo de salida convertida al formato IDMEF

La figura 4.3 contiene parte del código traducido al estándar IDMEF generado a la salida del sistema de APACS en uno de los diversos experimentos realizados.

4.2. Módulo de etiquetado cuantitativo

Un NIDS analiza el tráfico de la red paquete por paquete, y una vez lanzada una alerta, la información inicial de la que se dispone es la proporcionada por los propios campos del paquete. Además puede consultar la información correspondiente al detector, así como la regla que la ha activado, las características de la regla que la ha generado o las características generadas para el paquete analizado. El módulo clasificador de probabilidades tiene como objetivo principal determinar el grado en que una alerta ha sido emitida debido a una amenaza real. Para ellos se basa en la distancia entre las características que activan la alerta y las características generadas al analizar el paquete. El establecimiento de los distintos grupos de probabilidad se lleva a cabo mediante un proceso de agrupamiento previo a la etapa de detección. Este proceso está integrado en el entrenamiento del NIDS. Durante su desarrollo, se procede a la etiquetación de los grupos en función de los conjuntos de trazas de tráfico empleados en el entrenamiento del módulo detector. Una vez terminada esta fase, el sistema de correlación de alertas puede establecer de una manera eficiente el grupo al que pertenece la alerta generada por el paquete.

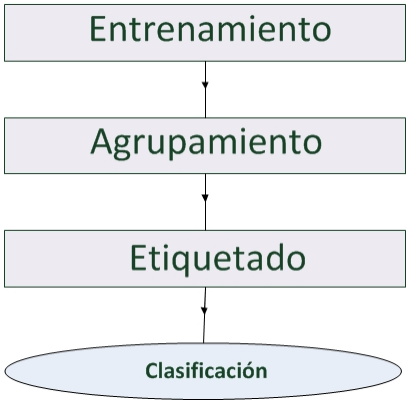

Figura 4.4: Módulo clasificador de probabilidades

En la figura 4.4 se muestra el esquema del módulo clasificador de las probabilidades de las alertas. En total se compone de las siguientes dos fases.

Fase 1: entrenamiento y creación de características del tráfico legítimo

Uno de los aspectos claves para que el NIDS funcione correctamente es la representación del espectro que genera el modo de uso habitual y legítimo de la red. En esta fase, el sistema inicializa y rellena las estructuras que representan el tráfico legítimo de la red que protege el NIDS. Esta etapa corresponde a la inicialización y

al entrenamiento base del NIDS. De hecho, utilizan las mismas estructuras.

Fase 2: agrupamiento y etiquetado

Para llevar a cabo el agrupamiento, ha sido modificado el entrenamiento del NIDS para que al generar las reglas, se cree un fichero con los valores de las características generadas para el paquete más representativo del ataque, y las características de referencia del tráfico legítimo obtenidas en el paso anterior. El objetivo es hallar el porcentaje de error entre ambos valores. De esta manera, es posible definir el grado en que se parecen, y es posible la creación de grupos conocidos como *clusters*, que puedan clasificarlos. Una vez finalizadas las fases del entrenamiento, no solamente se genera el conjunto de reglas necesarias para la detección, sino que se ha hallado los valores centrales de los grupos en que se va a clasificar el tráfico. Los grupos con los más bajos corresponden a aquellos que más se parecen al entrenamiento de referencias, es decir, al modo de uso habitual y legítimo de la red. Por el contrario, lo que tienen mayor porcentaje de error, tienen una mayor probabilidad de tratarse de verdaderas amenazas.

4.3. Módulo de etiquetado cualitativo

El objetivo del módulo de clasificación del tipo de ataque es definir cualitativamente la naturaleza de las alertas generadas por el NIDS. Para ello es necesario disponer de una base de conocimiento adecuada al tipo de amenazas con las que se va a lidiar. Si la base del conocimiento contiene una recopilación precisa y completa del contenido malintencionado propio del medio en que se lleva a cabo el despliegue, las anomalías podrían dejar de ser fluctuaciones del uso de la red respecto al modo de uso habitual legítimo y podrían pasan a ser ataques catalogados según el criterio que se considere apropiado. Es importante resaltar que el objetivo de la propuesta no abarca la elaboración de una compleja base del conocimiento sobre ataques. El objetivo es diseñar el cómo llevar a cabo dicha clasificación de una manera eficiente, en sincronía con las necesidades del NIDS.

A modo de sugerencia a la hora de elaborar dicha clasificación, en el año 2011 R. Hunt et al. [70] propusieron las pautas para llevar a cabo futuras taxonomías de ataques en sistemas informáticos, teniendo en cuenta factores cómo los sistemas a los que afectan, el tipo de red en que trabajan o por ejemplo las propias estrate-

gias de los ataques. Actualmente existen muchas clasificaciones completas, algunas generales como por ejemplo la taxonomía AVOIDIT [71] de S. Shiva et al. y otras específicas como por ejemplo la propuesta por Gruschka et al. [72] para ataques contra sistemas *Cloud*.

Antes de explicar cómo se ha aplicado la base del conocimiento en los experimentos realizados, es conveniente apuntar que para que la clasificación sea correcta, se ha de disponer de conjuntos de ataques para el entrenamiento correspondientes a cada una de las características que van a considerarse en el etiquetado. De esta manera, si por ejemplo para asociar una alerta a una clase es necesario que se cumplan dos propiedades, se debe disponer de tráfico representativo que permita identificar esas dos propiedades. Dichas colecciones de tráfico son utilizadas para generar distintos conjuntos de reglas, partiendo de una misma representación de tráfico legítimo.

Por su parte, la fase de generación de las reglas del NIDS para cada amenaza se va a llevar a cabo por separado. El objetivo de esto es conocer qué grupo de reglas se asocia con cada característica.

La figura 4.5 muestra el esquema de entrenamiento para la obtención de los conjuntos de reglas asociados a cada característica. Como puede verse, tanto la inicialización como el modelado del tráfico legítimo son comunes. A partir de entonces, para cada característica se lleva a cabo un entrenamiento en el que se extrae el conjunto de reglas producido por la colección de ataques que la contienen.

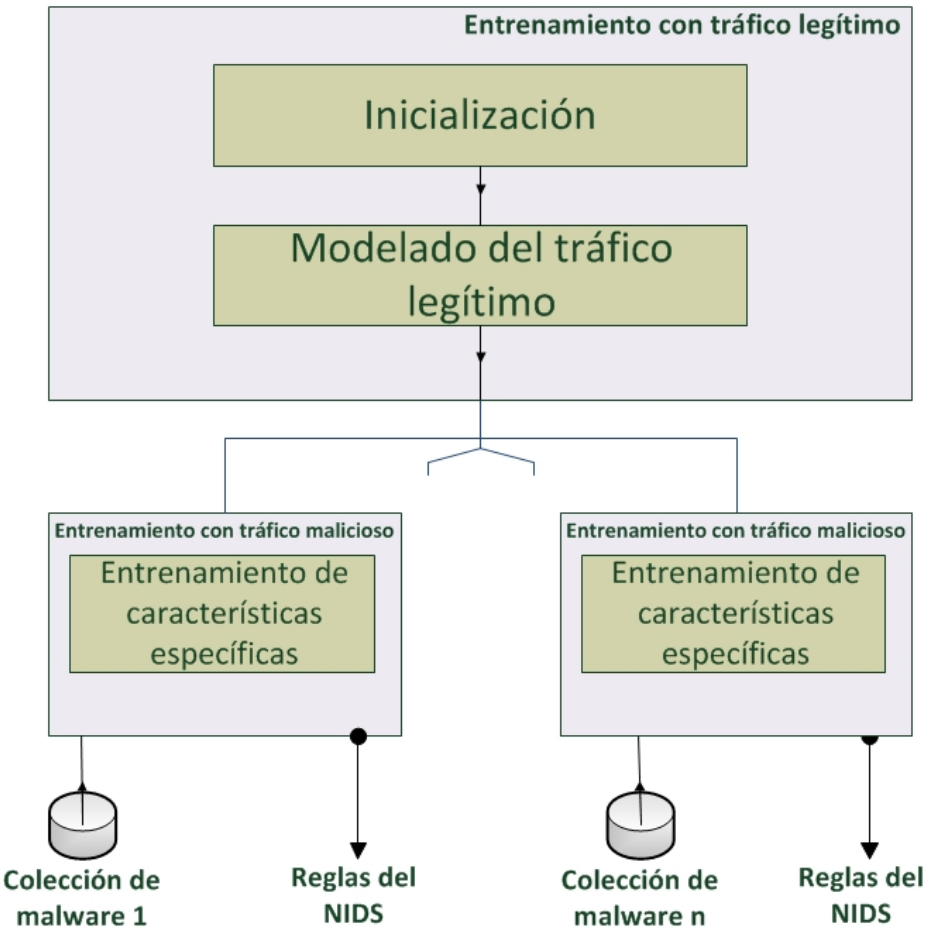

Figura 4.5: Esquema del entrenamiento del clasificador cualitativo

Finalmente se unifican las reglas generadas para cada característica. Cuando el NIDS analiza el contenido de un paquete, pueden activarse varias reglas a la vez. El módulo clasificador de probabilidades indica la probabilidad de que se traten de amenazas reales, y el módulo de clasificación de los ataques procesa cada una de las reglas activadas, con el fin de poder etiquetar la amenaza.

4.3.1. Módulo de clasificación a nivel de paquete

El módulo de clasificación a nivel de paquete tiene como objetivo establecer la clasificación de cada uno de los paquetes que atraviesan el NIDS, en función de las reglas que se activan en su fase de detección. Se trata de un módulo de respuesta rápida, y que bajo ninguna circunstancia debe penalizar el procesamiento en tiempo real del tráfico a analizar. Como posteriormente se explica, esta condición precisamente ha sido la que ha llevado a elegir una implementación mediante redes neuronales artificiales. Antes de explicar las características de la red neuronal que

lleva a cabo la clasificación y de profundizar en su entrenamiento, es importante entender la base de reglas que genera las clasificaciones en función de la base del conocimiento seleccionada.

Base de reglas

Hablar de una base de reglas cuando se ha comentado la decisión del uso de una red neuronal para establecer la presente clasificación puede parecer extraño. Sin embargo, detrás de la red neuronal se encuentra este mecanismo, ya que mediante su aplicación se ha generado las muestras que se emplean para establecer su conjunto de entrenamiento. Para la generación de la base de reglas se ha tenido en cuenta varias consideraciones iniciales:

- Los paquetes con contenido anómalo pueden activar más de una regla.
- Cada conjunto de reglas de los entrenamientos representa una característica
- El sistema ha de tener cierta tolerancia a errores. En consecuencia, se tiene en consideración que parte de las reglas que se activan al analizar un paquete pueden ser ruido.
- Un paquete está compuesto por uno o varios fragmentos. Normalmente el *malware* llega en varios fragmentos, debido a las restricciones de la unidad de trasnferencia máxima de la red MTU (*Maximum Transmission Unit*), por lo que existe la posibilidad de que la misma regla se active dos veces para el mismo paquete, pero en distintos fragmentos.

Las características que van a considerarse en la base de reglas, son conjuntos de reglas del NIDS representadas como el conjunto

$$Caracter\acute{i}sticas = \{C_0, C_1, C_2, \cdots, C_{n-1}\}$$

Cada una de ellas está definida por un grupo de reglas de detección. Para un mejor tratamiento, han sido ordenadas en función de la característica que representan. Sea m el total de reglas generadas por los distintos entrenamientos, se define el conjunto de reglas como

$$Reglas = \{R_0, R_1, R_2, \cdots, R_{M-1}\}$$

Por lo tanto cada característica C_i perteneciente al conjunto de características tal que $0 \leq i < n-1$, es definida por una subsección de reglas del conjunto de las

reglas. Si su regla inicial está en la posición p y su regla final está en la posición q, la característica C_i está definida como

$$C_i = \{R_p, R_{p+1}, R_{p+2}, \cdots, R_q\}$$

Conviene aclarar que puede darse el caso de que dos reglas sean iguales, ya que una anomalía puede tener más de una característica. Esta situación se soluciona generando varias reglas diferentes pero con el mismo contenido, cada una asociada al rango de cada una de sus características. Por último se definen las categorías como un conjunto

$$Categor\acute{i}as = \{E_0, E_1, E_2, \cdots, E_{T-1}\}$$

Las categorías son las diferentes clasificaciones que puede hacer la base de reglas. Al menos existe una categoría en cada base de reglas, la categoría "anomalía no definida"que es la clasificación por defecto de aquel contenido que no haya sido etiquetado de ninguna otra manera.

Ahora vamos a suponer que al módulo de detección llega un paquete. El análisis pasa por todas las reglas y si su carga útil alberga contenido anómalo saltarán una o varias de ellas. Además es posible que algunas de esas reglas se repitan en sus fragmentos, por lo que es necesario considerar la frecuencia de aparición de cada regla por paquete. El proceso de clasificación se lleva a cabo de la manera que se explica a continuación.

Paso 1

Se cuenta el número de veces que se activa cada regla al analizar el paquete

Paso 2

Se cuenta el número de veces que se han activado reglas correspondientes a cada característica. A ese valor le le denomina puntuación de las características. Formalmente se define de la siguiente manera: sea la característica C_i perteneciente al conjunto de características tal que $0 \leq i < n - 1$, estará definida por una subsección de reglas del conjunto de reglas. Si su regla inicial está en la posición p y su regla final está en la posición q, la característica C_i tal que

$$C_i = \{R_p, R_{p+1}, R_{p+2}, \cdots, R_q\}$$

se define la puntuación de C_i como P_i, tal que $0 \leq P_i < \infty$

$$P_i = \sum_{k=p}^{r} frec(K)$$

Siendo frec(K) el número de veces que se ha activado la regla que ocupa la posición K.

Paso 3

Una vez obtenida la puntuación de cada característica, se ordenan de mayor a menor. De esta manera se da predilección a aquellas características que tienen mayor puntuación. A la lista ordenada de ahora en adelante la llamaremos ORD, y para consultar la característica que ocupa cada una de sus posiciones se accede mediante $ORD(i)$, tal que $0 \leq i < n-1$. De esta manera el valor $ORD(0)$ contiene la característica de mayor puntuación, $ORD(1)$ contiene la característica con la segunda mayor puntuación y la característica $ORD(n-1)$ contiene la característica con la puntuación más baja.

Hay que tener cuidado con los casos en que dos características tengan la misma puntuación. Para ello se establece una tabla de desambigüaciones compuesta por cada una de las características y su peso asociado. El peso se asigna en función a la base del conocimiento elegida, y para que la tabla sea efectiva no puede haber dos características con el mismo peso. En el caso de que dos características tengan la misma puntuación, se consultan sus pesos en dicha tabla. Al insertarse en ORD se colocaran en las primeras posiciones las características con mejor peso, solucionando así el conflicto.

Paso 4

Una vez generada la lista ordenada ORD basta con consultar la base de reglas para establecer una clasificación. La decisión se lleva a cabo considerando las características más representativas, es decir, la de mejor puntuación.

Número	Regla	Clasificación
1	Si $Ord(0) = C_o$ Y $Ord(1) = C_1$	E_o
2	Si $Ord(0) = C_1$ Y $Ord(1) = C_0$	E_o
3	Si $Ord(0) = C_o$ Y $Ord(1) = C_2$ Y $Ord(2, \cdots, n-1) = \emptyset$	E_1
4	Si $Ord(0) = C_2$ Y $Ord(1) = C_0$ Y $Ord(2, \cdots, n-1) = \emptyset$	E_1
5	Si $Ord(0) = C_2$ Y $Ord(1, \cdots, n-1) = \emptyset$	E_2
6	Si $Ord(0) = C_1$ Y $Ord(1, \cdots, n-1) = \emptyset$	E_3
7	Si $Ord(0) = C_o$ Y $Ord(1) = C_3$	E_4
8	Si $Ord(0) = C_3$ Y $Ord(1) = C_0$	E_4
9	Si $Ord(0) = C_3$ Y $Ord(1, \cdots, n-1) = \emptyset$	E_5
10	Si $Ord(0) = C_1$ Y $Ord(1) = C_3$	E_6
11	Si $Ord(0) = C_3$ Y $Ord(1) = C_1$	E_6
12	$Else$	E_7

Tabla 4.1: Base de reglas a nivel de paquete

La tabla 4.1 muestra un ejemplo de base de reglas que considera las dos características más representativas del paquete y establece una clasificación. En este ejemplo se definen:

$$Caracter\acute{i}sticas = \{C_0, C_1, C_2, C_3\}$$

$$Categor\acute{i}as = \{E_0, E_1, E_2, E_3, E_4, E_5, E_6, E_7\}$$

Para cada una de las cuatro características se ha hecho un entrenamiento específico y se ha generado su conjunto de reglas. Para establecer una clasificación basta con aplicar la regla correspondiente. En el ejemplo hay reglas que requieren de dos características e ignoran el valor del resto $(1, 2, 7, 8, 10, 11)$ y otras que requieren que el resto de característica haya tenido una puntuación nula $(3, 4, 5, 6, 8)$. Algunas tienen en consideración más de una característica, y la última regla es la que se activa por defecto estableciendo la etiqueta E_7 en aquellos paquetes cuyo contenido no se haya podido identificar.

Como puede verse, la generación de la base de reglas depende directamente de la base del conocimiento. De esta manera, si por ejemplo la característica C_0 corresponde a "presencia de técnicas de ofuscación de *malware*", y la característica C_1 a "presencia de técnicas de desbordamiento de *buffer*", la regla 1 de la tabla puede traducirse de forma natural como "si la mayor parte de reglas del NIDS que han saltado corresponden con presencia de *malware* ofuscado y presencia de técnicas de desbordamiento de *buffer* entonces su etiqueta es E_0". Donde la clasificación E_0 podría indicar "intento de explotación de desbordamiento de pila a través malware

ofuscado".

Si Además la característica C_3 fuera "presencia de mecanismo de control remoto", las reglas 7 y 8 pueden traducirse a lenguaje normal como "Si la mayor parte de las reglas del NIDS que han saltado corresponden con la presencia de malware ofuscado y mecanismos de control remoto, entonces su etiqueta es E_4". E_4 podría indicar "presencia de trojano".

Clasifiación mediante redes neuronales artificiales

Una vez establecida la base de reglas se procede al diseño de la red neuronal artificial. Los parámetros de entrada corresponden con cada una de las características que se emplean en la base de reglas para establecer las clasificaciones, es decir, tiene n entradas correspondientes a $C_0, C_1, C_2, \cdots, C_{n-1}$ características. Cada una de ellas acepta un rango de valores v tales que $0 \leq v < 1$, indicando el porcentaje de la puntuación total que representan

$$v = \frac{P_i}{\sum_{k=0}^{n-1} P_k}$$

El valor de salida representa hasta T valores coincidiendo con las $E_0, E_1, E_2, \cdots, E_{T-1}$ posibles clasificaciones. La salida tiene un valor S tal que $0 \leq S < 1$. En la fase de entrenamiento es posible asignar un rango de salida distinto a cada etiqueta.

Clasificación	Valor
E_O	0.0
E_1	0.1
E_2	0.2
E_3	0.3
E_4	0.4
E_5	0.5
E_6	0.6
E_7	0.7

Tabla 4.2: Etiquetados para la clasificación de paquetes

En la tabla 4.2 se puede ver un ejemplo de salidas asociadas a las clasificaciones de la base de reglas mostrada en la tabla 5.1. El resto de características elegidas para la red neuronal del ejemplo vienen resumidas en la tabla 4.3.

Característica
Número de Entradas: T
Número de Salidas: 1
Número de Capas: 3 (1 oculta)
Porcentaje de error: 0.001
Número Máximo de etapas: 50000000
Número de Capas Ocultas: 1
Función de Activación: Elliot

Tabla 4.3: Parámetros de configuración de la red neuronal

A excepción de las condiciones de entrada y salida de la red neuronal, tanto la topología como el resto de consideraciones de diseño pueden adaptarse a las necesidades de la base de reglas. Como ya se mencionó, el entrenamiento de la red resulta largo y tedioso, así que es recomendable establecer una red que permita alcanzar el error deseado en un tiempo razonable, y que sea capaz de dar la suficiente precisión al clasificador. En la red neuronal del ejemplo se ha optado por usar la función de activación Elliot implementada en las librerías FANN para C con las que se desplegó la red. El motivo fue empírico. El proceso de entrenamiento era aparentemente mucho más rápido que con las funciones de activación clásicas como la lineal o la sigmoidal.

```
F.Elliot  span: 0 < y < 1
y = ((x*s) / 2) / (1 + |x*s|) + 0.5
d = s*1/(2*(1+|x*s|)*(1+|x*s|))
```

Los resultados obtenidos fueron bastante buenos, con un porcentaje de acierto del 96,7 % en la etiquetación. No obstante, las características del diseño se deben elegir en función de los requisitos del sistema a desplegar.

Entrenamiento de la red neuronal artificial

Otro de los aspectos importantes a tener en cuenta sobre el clasificador de paquetes, es el proceso de entrenamiento. Será precisamente esta fase la que decida lo bien o mal que va a funcionar el sistema. En el entrenamiento se utiliza directamente la base de reglas. Cada una de las muestras del entrenamiento es un vector de n posiciones que representan las n puntuaciones de las n características y un valor de salida que representa la clasificación que debe tener. El proceso de generación de las muestras se lleva a cabo en cuatro pasos.

Paso 1

Se elige aleatoriamente una posición del vector y se le asigna un valor entero aleatorio mayor que 0 y menor que una cota establecida previamente (en el ejemplo se estableció en 50). De esta manera es posible asegurarse que la muestra al menos contenga representación de una característica.

Paso 2

Para el resto de posicione del vector se decide su representación mediante la generación de un número aleatorio. Si el numero generado es inferior a un factor de aparición también pre-establecido (en el ejemplo se estableció en 0.15), tendrá representación. Al igual que en el paso 1, en este caso se le asigna un valor de aleatorio que represente su puntuación.

Paso 3

Se sustituye cada posición del vector por el porcentaje de la puntuación total que representa.

Paso 4

Aplicando la base de reglas, se establece una clasificación del vector. Como anteriormente se mencionó, es necesario ordenarlo de mayor a menor, y se ha de contar con la correspondiente tabla de desambigüación en función de la base del conocimiento. El ordenamiento de mayor a menor se considera únicamente para decidir la salida que el clasificador debe generar. Los valores del vector de cada una de las muestras permanecen como resultan del paso 3, ahorrando el coste del algoritmo de ordenamiento. Cada posición del vector va a ser la entrada de cada una de las entradas de la red neuronal en el entrenamiento. El valor asociado a la clasificación es la salida deseada.

Una vez generado un conjunto de muestras lo suficientemente grande, se pasa al proceso de entrenamiento. Como en cualquier red neuronal, se calibran los pesos de las neuronas mediante iteraciones sobre las muestras hasta que las salidas generen un error inferior al establecido (0.001 en el ejemplo) o hasta haber concluido en número de iteraciones máximas. En ese momento concluye la fase de entrenamiento.

Experimento local

Para clasificar el tráfico con la red neuronal, basta con generar un vector de n entradas, donde n es el número de categorías que se consideran para la etiquetación. De esta manera

$$V = \{V_0, V_1, ..., V_2, V_3\}$$

Donde cada V_i tal que $0 \leq i < n$ representa la frecuencia de aparición normalizada de reglas del conjunto de la característica C_i que han activado una alerta en la fase de detección. Una vez introducidos los datos, la red neuronal devolverá un valor z, tal que $0 \leq z < 1$, que identifica la etiqueta correspondiente a la clasificación de la anomalía.

Ejemplo de aplicación

Para probar la tasa de aciertos de las redes neuronales implementadas, se ha tomado como referencia una generación de muestras similar a la del entrenamiento. La base del conocimiento de los experimentos se manifiesta en la base de reglas que en el apartado anterior se mostraron como ejemplo.

Característica	Significado
C_O	Presencia de técnicas de ofuscación de *malware*
C_1	Presencia de técnicas de desbordamiento de *buffer*
C_2	Actividad anómala relacionada con *Cloud Computing*
C_3	Presencia de mecanismo de control remoto

Tabla 4.4: Características del experimento a nivel de paquete

Etiqueta	Significado
E_O	Desbordamiento de pila a través de malware *ofuscado*
E_1	Infiltración de *malware* ofuscado en sistema *Cloud*
E_2	Actividad anómala relacionada con *Cloud*
E_3	Técnicas de desbordamiento de *buffer*
E_4	Control remoto mediante troyano
E_5	Mecanismos de control remoto
E_6	Escalada de privilegios para control remoto
E_7	Anomalía no identificada

Tabla 4.5: Clasificaciones del experimento a nivel de paquete

Número	Regla	Clasificación
1	Si $Ord(0) = C_o$ Y $Ord(1) = C_1$	E_o
2	Si $Ord(0) = C_1$ Y $Ord(1) = C_0$	E_o
3	Si $Ord(0) = C_o$ Y $Ord(1) = C_2$ Y $Ord(2, \cdots, n-1) = \emptyset$	E_1
4	Si $Ord(0) = C_2$ Y $Ord(1) = C_0$ Y $Ord(2, \cdots, n-1) = \emptyset$	E_1
5	Si $Ord(0) = C_2$ Y $Ord(1, \cdots, n-1) = \emptyset$	E_2
6	Si $Ord(0) = C_1$ Y $Ord(1, \cdots, n-1) = \emptyset$	E_3
7	Si $Ord(0) = C_o$ Y $Ord(1) = C_3$	E_4
8	Si $Ord(0) = C_3$ Y $Ord(1) = C_0$	E_4
9	Si $Ord(0) = C_3$ Y $Ord(1, \cdots, n-1) = \emptyset$	E_5
10	Si $Ord(0) = C_1$ Y $Ord(1) = C_3$	E_6
11	Si $Ord(0) = C_3$ Y $Ord(1) = C_1$	E_6
12	$Else$	E_7

Tabla 4.6: Base de reglas del experimento a nivel de paquete

Como puede verse en las tablas 4.4, 4.5 y 4.6, se trata de una pequeña base del conocimiento para probar el sistema que habría que ampliar considerablemente para su uso en entornos reales. Para llevar a cabo las pruebas se ha diseñado una herramienta parecida a la empleada para la generación del conjunto de entrenamiento. Dicha herramienta genera una serie de muestras aleatorias, y comprueba si el resultado que genera la red neuronal coincide con el que genera la base de reglas.

En total se probaron 50.000 muestras obteniendo una tasa de acierto del 96,7 %. La red neuronal implementada coincide con la explicada a modo de ejemplo en la subsección anterior, y variaciones sobre ella podrían producir un mejor comportamiento. Los resultados del experimento han sido satisfactorios, aunque con un ajuste mejor de la red neuronal, podrán llegar a ser aún mejores. Es importante considerar que con la configuración del ejemplo, despues de muchas horas de entrenamiento no se alcanzó el porcentaje de error del 0.001 %, deteniendose el proceso de entrenamiento al llegar al máximo número de iteraciones. Los resultados se muestran en la figura 4.6

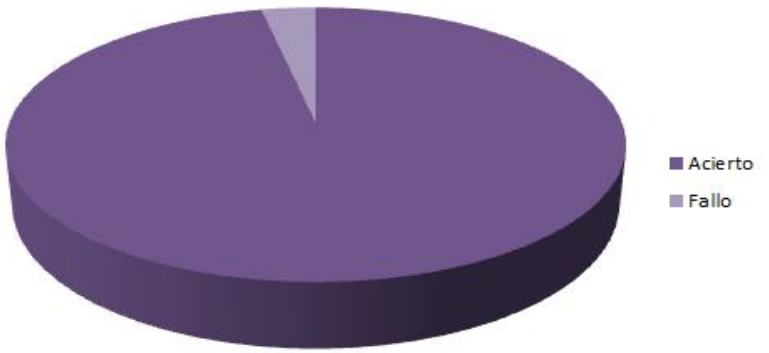

Figura 4.6: Resultados del experimento a nivel de paquete

4.3.2. Módulo de clasificación a nivel de traza

En un primer diseño, el módulo de correlación de alertas lo integraban tan solo los dos módulos anteriores. Esa decisión completaba el modelo híbrido, cubriendo de esta manera las necesidades de clasificación en tiempo real. Pero a la hora de revisar el trabajo, aparecieron varios aspectos nuevos y relevantes que era necesario considerar.

- El primero de ellos es que el NIDS analiza la carga útil del tráfico analizado, y por lo tanto, está especializado en la detección de *malware*. La reconstrucción del escenario es apropiada para la identificación de ataques distribuidos, pero en este caso no es la mejor opción.

- En consecuencia, es necesario destacar la necesidad de obtener una clasificación a nivel de traza. El contexto en el que se ha detectado un paquete anómalo, o un conjunto de paquetes anómalos incluye información valiosa que es conveniente considerar.

- Por otro lado, el estudio forense de las trazas de un ataque, debe llevar información a nivel de paquete y a nivel de traza. De esta manera el operador puede conocer el tipo de amenaza que han comprometido sus sistemas, y comprobar paquete a paquete su composición.

- Algunas técnicas de ataque, como las de denegación de servicio [74] o las de reconstrucción de paquetes conocidas como *packet crafting* [75], se detectan con mayor facilidad considerando los paquetes como un conjunto en lugar de individualmente.

- La extracción y etiquetación de trazas de ataques maliciosos, permiten generar conjuntos de entrenamiento con los que el NIDS puede ser entrenado. De esta manera, es posible fortalecerlo frente a futuros ataques similares.

A vista de estos problemas, se ha decidido añadir un módulo clasificador a nivel de trazas al sistema. Antes de continuar es importante definir lo que de ahora en adelante va a considerarse como traza. Para APACS el concepto de traza se define como un conjunto de paquetes que circulan a lo largo de una red durante un intervalo de tiempo concreto.

A partir de esta definición surge un nuevo problema: la decisión del periodo de tiempo que va a delimitar las trazas que van a ser analizadas por el sistema de correlación de alertas. Si el uso de dicho sistema corresponde a un análisis forense, entonces el problema se reduce; El estudio comienza cuando el sistema ha sido comprometido.

Pero para un clasificador en tiempo real, como es el caso de APACS, no se trata de un problema trivial. Aunque se haya podido detectar un intento de intrusión, es complicado establecer el punto preciso de corte. La decisión tomada a la hora de implementar el sistema de correlación de alertas es dividir el tráfico en secciones enmarcadas en un mismo intervalo de tiempo o en un mismo número de paquetes. De esta manera, el tráfico queda seccionado en conjuntos de paquetes de dimensiones parecidas. APACS etiqueta cada una de esas secciones de acuerdo a una base del conocimiento parecida a la del apartado anterior, pero correspondiente a amenazas que puedan darse distribuidas en varios paquetes.

Este tipo de segmentación puede dificultar el acierto del clasificador, así como proveer al atacante de la capacidad de usar técnicas de evasión [76] para dificultar el proceso de etiquetado. En el orden de evitar este tipo de amenazas, una condición esencial del diseño es que el etiquetado sea no determinista. Con un etiquetado no determinista, el clasificador genera diferentes posibles clasificaciones para la traza en función de patrones parecidos a su contenido. Para que esta condición no obstruya al operador que analice los resultados, se provee de un mecanismo que calcula la probabilidad de que la etiqueta correcta sea cada una de las propuestas. Para poder generar ese no determinismo, y calcular las probabilidades de acierto, el diseño se ha llevado a cabo mediante la implementación de un algoritmo genético.

Posiblemente esta decisión pueda crear controversia. La elección de un algoritmo genético no parece una decisión acertada para un sistema que trabaja en tiempo real. Sin embargo, la sencillez del algoritmo hace que la penalización en base a su coste del tiempo de ejecución sea apenas significativa respecto a la del resto del sistema. Por otro lado, esta penalización se produce cada cierto periodo de tiempo, es decir, entre traza y traza analizadas. Esto refuerza el hecho de que su impacto sea muy bajo. A efectos globales, es posible afirmar que este pequeño retraso es admisible de cara a la información detallada que el módulo va a generar. Además si la amenaza es muy clara y directa, serán los otros módulos quienes se encarguen de clasificarla a tiempo para forzar su interrupción.

Algorítmo genético

El módulo de clasificación a nivel de trazas consiste en un algoritmo genético básico [77]. Dichos algoritmos comenzaron a aplicarse como solución a problemas de cómputo entre los años 60 y 70 [78] [79], y posteriormente también se ofrecen como alternativa a estrategias de clasificación basadas en minería de datos [80]. Este tipo de algoritmo tiene como característica el efectuar la búsqueda de su solución en base a la probabilidad. De esta manera, si el algoritmo está bien configurado, la solución tiene tendencia a converger en la denominada solución óptima.

Este tipo de algoritmos hacen evolucionar una población de individuos generada a partir de los datos de entrada, sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas). A continuación se lleva a cabo a una selección de acuerdo con algún criterio, en función de la cual se decide los individuos más adaptados que van a sobrevivir, y los menos aptos que serán descartados. En función de las características de esos individuos se construye la solución.

El algoritmo genético mediante el que se lleva el análisis de las alertas generadas a nivel de traza recibe como entrada las salidas del módulo de clasificación a nivel de paquetes explicado en la sección anterior. Basándose en ellas, elabora una población inicial, y extrayendo los patrones representativos de cada individuo, tratará de que encajen con los patrones correspondientes a una base del conocimiento relacionada con trazas de ataques en lugar de paquetes. Si no existe un número importante de individuos cuyos patrones coincidan, se realizan nuevos pasos evolutivos atendiendo al esquema general de los algoritmos genéticos básicos. A continuación se exponen los aspectos significativos del algoritmo genético implementado.

Población y genotipo

Un individuo es definido como un vector de dimensión d que representa su genotipo.

$$Genotipo = \{C_0, C_1, \cdots, C_{d-1}\}$$

Cada valor C_i, $0 \leq i < d$ corresponde al gen que ocupa la posición i del vector. A priori no va a establecerse un valor adecuado para d, ya que va a depender de la base de reglas y otras cuestiones de diseño. El parámetro d elegido para los experimentos tiene asociado el valor 6. La elección del mismo se basa en dos aspectos: por un lado, cuanto menor sea d, menor riqueza tendrá el genotipo de los individuos. Esto conlleva una menor diversidad poblacional y por lo tanto deja más al azar el número de pasos necesarios para la convergencia, también incrementado las posibilidades de error.

En el otro lugar, la elección de valores excesivamente altos obstruyen la capacidad del cómputo de APACS en tiempo real. Debido a estos dos aspectos, es conveniente elegir un valor lo más alto posible que no comprometa el rendimiento del sistema.

Para rellenar el genotipo se va a crear un conjunto con todas las clasificaciones que el módulo de clasificación de paquetes ha generado a lo largo de la traza a etiquetar. En este conjunto se puede repetir clasificaciones, de forma que aquellas que han aparecido más veces van a tener mayor representación. Cada gen del genotipo tiene un elemento de dicho conjunto asignado aleatoriamente. Esto da mayor probabilidad de aparición a las etiquetas más repetidas en la traza, y un cierto grado de aleatoriedad que otorga riqueza poblacional.

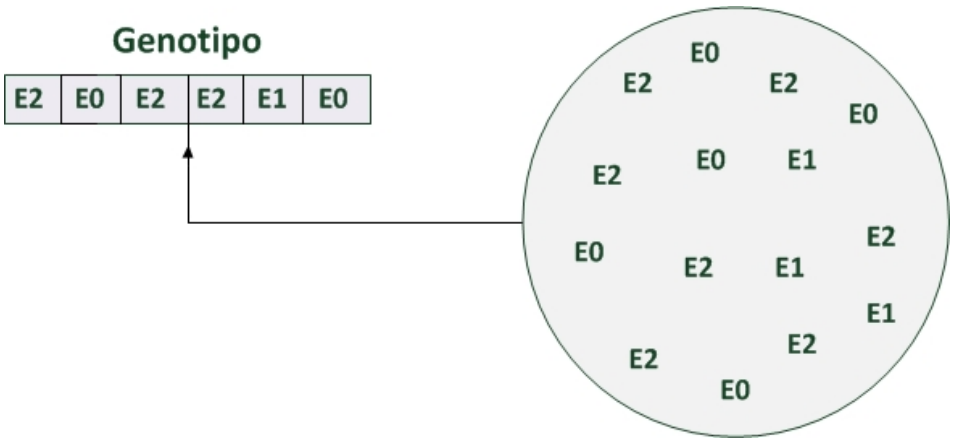

Figura 4.7: Ejemplo de generación de genotipo

La figura 4.7 muestra un ejemplo de generación del genotipo de uno de los individuos de la población. En él, un conjunto situado en la parte derecha representa las distintas etiquetas generadas por el módulo de clasificación a nivel de paquetes, y el vector de la derecha, un posible rellenado extrayendo aleatoriamente algunos de ellos.

En un algoritmo genético, el concepto de población se define como conjunto de individuos que participan en la búsqueda de la solución óptima. En el diseño, se habla de tres tipos de poblaciones

Población inicial

La población inicial es el conjunto de individuos que se genera inicialmente a partir de los datos de entrada. Es integrada por un número mayor de representantes que la población de trabajo, ya que previamente al inicio del propio algoritmo en sí, se establece una selección de sus mejores individuos para formar parte de ella.

Población de trabajo

La población de trabajo inicialmente es constituida por los mejores individuos de la población inicial. Es la que interviene directamente durante la ejecución del algoritmo genético y sobre la que se producen las operaciones de cruce y mutación.

Población élite

La población élite es el subconjunto de los mejores individuos de la población inicial. Se utiliza para determinar la solución óptima, que como se explica más adelante, tiene como condición que todos sus componentes hayan finalizado con éxito su encaje con algún patrón correspondiente a alguna etiqueta de la base del conocimiento.

Patrones asociados a genotipos

El módulo de análisis a nivel de trazas considera como patrón a un vector representativo de alguna de las posibles clasificaciones que pueda generar. Partiendo de la definición de patrón

$$Patrón = \{Ca_0, Ca_1, \cdots, Ca_{T-1}\}$$

y considerando esta vez T como el número total de características que son consideradas en la base del conocimiento, se denomina a Ca_i tal que $0 \leq i < d$ como la característica que ocupa la posición i-esima de una ordenación mayor a menor de las

frecuencias de aparición de las características asociadas al genotipo de un individuo.

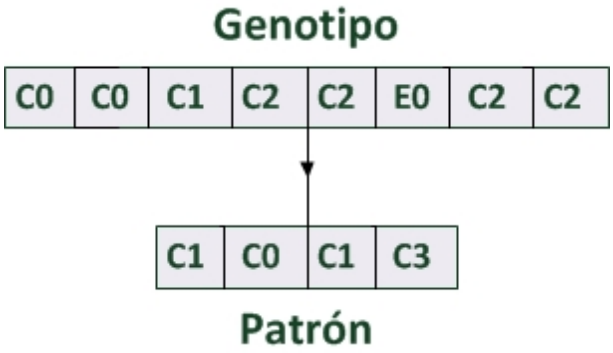

Figura 4.8: Ejemplo de obtención del patrón de un genotipo

La figura 4.8 muestra un ejemplo de obtención del patrón de un genotipo. En este caso, el conjunto de características es $C = \{C_0, C_1, C_2, C_3\}$. Se supone que el genotipo se ha rellenado a partir de las clasificaciones llevadas a cabo para cada paquete como se indica en la subsección inmediatamente anterior. El patrón que produce es un vector que contiene todas las características consideradas, y que se encuentra ordenado de mayor a menor dependiendo de su frecuencia de aparición. De esta forma, la característica más repetida es C_2 con una frecuencia de aparición de 4, seguida de C_0 con una frecuencia de aparición de 3, seguida por C_1 con una frecuencia de aparición de 1, y finalmente C_3 con una frecuencia de aparición de 0.

A la vista del uso de este tipo de patrones, vuelve a surgir un problema cuya solución también se explica en la sección en que se define la base de reglas para el módulo de clasificación a nivel de paquetes: la ambigüedad. La decisión del orden de inserción en el patrón de dos características con la misma frecuencia de aparición se resuelve fácilmente mediante el uso de una tabla de desambigüación. Dicha tabla ha de plantearse en la fase de diseño, y es totalmente dependiente de la base del conocimiento en que se basen las clasificaciones.

La tabla 4.7 muestra un ejemplo de los patrones asociados a cada una de las reglas de una posible base del conocimiento de anomalías a nivel de traza. En ella se tiene en consideración 4 características $c = \{C_0, C_1, C_2, C_3\}$. Los caracteres '-' indican que puede tratarse de cualquier valor y el carácter '$\emptyset$' que no puede haber apariciones. Aunque esta base de reglas se parezca mucho a la del ejemplo del módulo de clasificación de paquetes, no hay que confundirse: sus bases del conocimiento son

Número	Regla	Patrón	Clasificación
1	Si $Ord(0) = C_2$ Y $Ord(1) = C_1$	$[2, 1, -, -]$	E_o
2	Si $Ord(0) = C_1$ Y $Ord(1) = C_2$	$[1, 2, -, -]$	E_o
3	Si $Ord(0) = C_0$ Y $Ord(1) = C_2$	$[0, 2, -, -]$	E_1
4	Si $Ord(0) = C_2$ Y $Ord(1) = C_0$	$[2, 1, -, -]$	E_1
5	Si $Ord(0) = C_2$ $YOrd(1, \cdots, n-1) = \emptyset$	$[0, \emptyset, \emptyset, \emptyset]$	E_2
6	Si $Ord(0) = C_1$ $YOrd(1, \cdots, n-1) = \emptyset$	$[1, \emptyset, \emptyset, \emptyset]$	E_3
7	Si $Ord(0) = C_0$ Y $Ord(1) = C_3$	$[0, 3, -, -]$	E_4
8	Si $Ord(0) = C_3$ Y $Ord(1) = C_0$	$[3, 0, -, -]$	E_4
9	Si $Ord(0) = C_3$ $YOrd(1, \cdots, n-1) = \emptyset$	$[3, \emptyset, \emptyset, \emptyset]$	E_5
7	Si $Ord(0) = C_1$ Y $Ord(1) = C_3$	$[1, 3, -, -]$	E_6
8	Si $Ord(0) = C_3$ Y $Ord(1) = C_1$	$[3, 1, -, -]$	E_6
9	otras	$[-, -, -, -]$	E_7

Tabla 4.7: Base de reglas del experimento a nivel de traza

totalmente distintas. Una considera características y asociaciones en función de una base del conocimiento a nivel del contenido de un paquete, y ésta lo hace a partir del contenido de un conjunto de paquetes, lo que implica que las características y las amenazas sean muy diferentes.

Función de aptitud y solución óptima

En un algoritmo genético la función de aptitud tiene como objetivo evaluar cuanto de bueno o de apto para el cruce es un individuo. A partir de ella se va a definir el concepto de solución óptima, por lo que es conveniente entender su comportamiento. Para calcular la aptitud de cada individuo, se lleva a cabo una comparación entre el patrón asociado al genotipo de dicho individuo, y una serie de patrones conocidos establecidos a partir de una base de conocimiento similar a la explicada para el módulo clasificador a nivel de paquetes. La medida en que se lleva a cabo la comparación es a través de la distancia de Levenshtein.

La distancia de Levenshtein se usa habitualmente en las ciencias de la computación para calcular el grado de similitud entre dos vectores, que es lo que son los dos patrones. En el año 1965 se definió [81] para calcular el número de pasos necesarios para convertir una cadena de caracteres en otra mediante operaciones de inserción, eliminación o sustitución. Se la considera una generalización de la conocida distancia de Hamming [82] comúnmente empleada para la comprobación del número de errores en envíos por redes. Pero al contrario que la distancia Levenshtein, la distancia de Hamming únicamente considera la operación de sustitución, lo que es más que

suficiente para llevar a cabo un conteo a nivel de *bits*.

Cuando la función de aptitud devuelve el valor 0, no es necesario realizar operaciones sobre el primer vector para transformarla en el segundo vector, debido a que desde un primer momento se trata de vectores idénticos. El primer vector es el patrón asociado al genotipo del individuo y el segundo es el patrón asociado a alguna de las clasificaciones posibles. Para calcular el valor de aptitud se calcula la distancia Levenshtein entre el patrón del individuo y cada una de las clasificaciones, y se devuelve el menor de los resultados obtenidos, es decir, la mejor comparación. Considerando este punto, la solución óptima va a ser aquella en que la población élite al completo tenga un valor de aptitud 0, es decir, que todos sus individuos coincidan con al menos uno de los patrones.

Operadores de cruce y mutación

La decisión del comportamiento de los operadores de cruce y mutación debe depender del resto de características de diseño, como por ejemplo la longitud d del genotipo o el número de características que tiene en consideración la base del conocimiento. También es condicionada por las características que se esperen del sistema, como el grado de precisión o la diversidad de las soluciones. En lo referente a estos aspectos, se detalla los operadores implementados para los experimentos realizados dando pie a que futuros trabajos replanteen el uso de los mismos.

La operación de cruce en la implementación consiste en establecer aleatoriamente una posición pivote común a los genotipos de los dos individuos que intervienen, e intercambiar sus extremos. Como resultado, el primer individuo conserva los genes de su extremo izquierdo, pero su extremo derecho corresponde a los genes del otro individuo antes del cruce. Con los otros dos extremos ocurre precisamente el caso opuesto. De esta manera los genotipos de los vectores producen un intercambio a partir del pivote de sus genotipos originales.

La operación de mutación recorre los distintos genes de un individuo. Para cada uno de ellos existe una probabilidad denominada probabilidad de mutación que decide si el gen permanece en su estado original o es sustituido por una característica al azar proveniente del conjunto con el que se generaron los genes iniciales, y que como se indica en una subsección anterior, recolecta las clasificaciones generadas para cada uno de los paquetes que integran la traza.

Algorítmo genético

El algoritmo genético de APACS comienza con la creación de una población inicial a partir de la etiquetación establecida para cada uno de los paquetes que componen la traza a analizar, previamente efectuada por el módulo clasificador a nivel de paquetes. De esta población inicial se elegirá a los mejores individuos, es decir, aquellos para los que la función de aptitud ha producido las salidas más bajas. Estos individuos integran la población de trabajo, que es la que se tiene en cuenta durante el resto del algoritmo.

La manera de avanzar hacia la solución consiste en ir iterando una serie de operaciones de cruce y mutación hasta que se alcance la solución óptima o hasta que se haya realizado el número máximo de iteraciones indicado. La importancia de este último parámetro es esencial, ya que en los casos en que la solución óptima sea inalcanzable, su valor va a determinar la precisión de la aproximación a dicha solución.

La operación de cruce se lleva a cabo en un individuo si al calcular un valor aleatoriamente, ha superado el porcentaje indicado por la probabilidad de cruce. De esta manera, la probabilidad de cruce indica la probabilidad de que cada individuo de la población de trabajo efectúe la operación de cruce. El otro individuo que participa en ella se elige azarosamente de entre el resto de individuos. Una vez que se ha producido un cruce, existe una posibilidad de que el resultado mute. La probabilidad de mutación indica el riesgo de que se produzca dicha mutación.

Una vez finalizado el algoritmo, se procede al tratamiento de los resultados. Para ello se recorre toda la población en busca de individuos cuyo genotipo haya producido patrones que encajen con los de la base del conocimiento. La frecuencia de encaje con cada patrón indica la probabilidad de que la traza del tráfico coincida con la etiqueta asociada a dicho patrón.

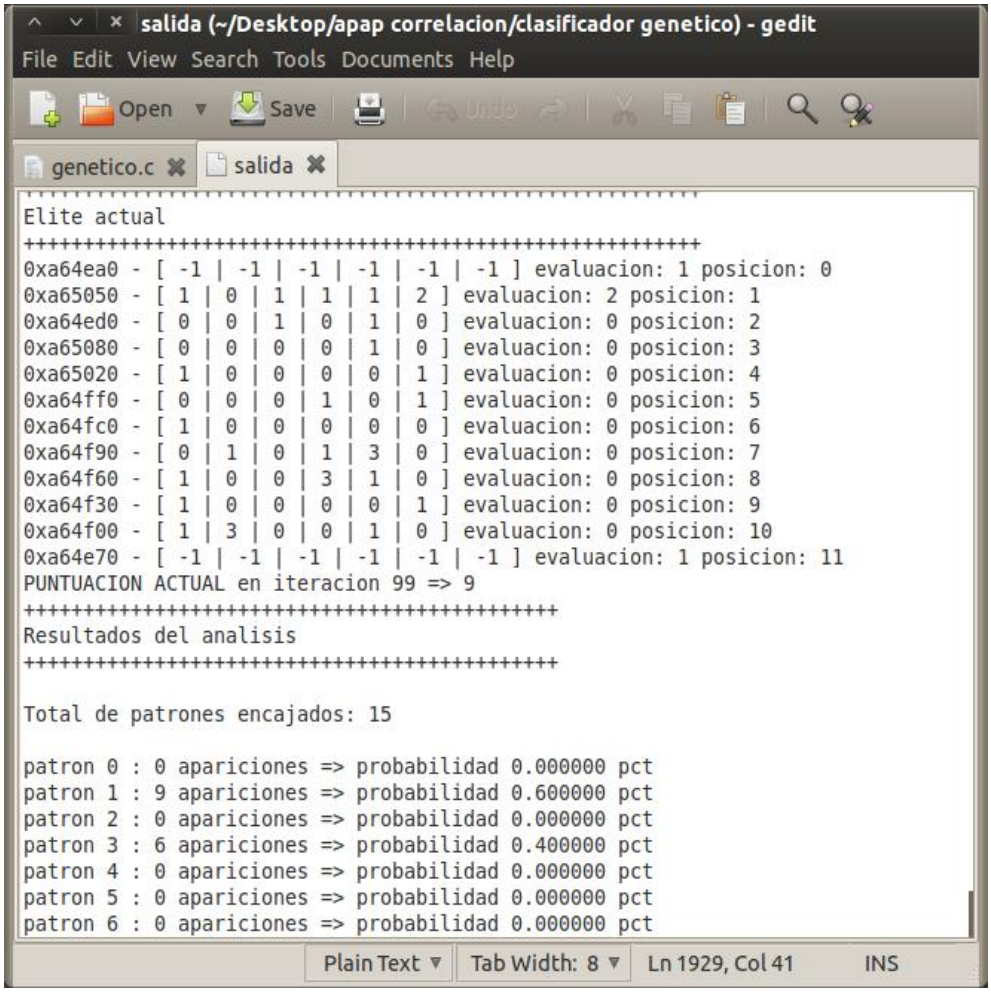

Figura 4.9: Ejemplo de clasificaciones mediante el algoritmo genético

La figura 4.9 muestra un ejemplo de los resultados generados en uno de los experimentos. En esa prueba, la clasificación no estaba clara y el sistema ha propuesto un 40 % de probabilidades de que corresponda con la etiqueta asociada al patrón 3 y un 60 % de que lo haga con la del patrón 1. En una clasificación determinista se habría etiquetado únicamente correspondiendo con el patrón de mayor frecuencia de aparición, lo que provocaría una pérdida de información importante de cara a un análisis forense. El listado inferior contiene los parámetros que van a determinar el comportamiento del algoritmo genético de APACS y por lo tanto la calidad de las salidas que genera.

Longitud del genotipo Es el valor d. Cuanto mayor sea su valor, aporta mayor diversidad genética, pero peor es el rendimiento.

Longitud de población inicial Cuanto mayor sea la población inicial, mejor será la población de trabajo, pero peor es el rendimiento.

Longitud de población de trabajo Cuanto mayor sea la población de trabajo, aporta mayor diversidad genética, pero peor es el rendimiento.

Longitud de población elite Cuanto más grande sea la población elite, más completa va a ser la salida del clasificador, ya que puede incluir más posibles etiquetas. Además proporciona mayor precisión, pero conlleva un mayor número de iteraciones para alcanzar la solución óptima, penalizando de esta manera el rendimiento.

Probabilidad de cruce Cuanto mayor sea su valor, mayor será la diversidad genética. Valores excesivamente altos pueden disminuir la precisión del sistema, generando una salida con mayor grado de aleatoriedad.

Probabilidad de mutación Cuanto mayor sea su valor, mayor será la diversidad genética. Valores excesivamente altos pueden disminuir la precisión del sistema, generando una salida con mayor grado de aleatoriedad.

Número máximo de iteraciones Cuanto más alto mejor será la solución pero existirá una probabilidad mayor de penalizar el rendimiento.

Ejemplo de aplicación

Para llevar a cabo las evaluaciones se han generado aleatoriamente distintos conjuntos de supuestas clasificaciones para los paquetes que integran las trazas. La tabla 4.8 muestra los parámetros asignados al algoritmo.

Parámetro	Valor
Longitud del genotipo	6
Longitud de población inicial	200
Longitud de población de trabajo	50
Longitud de población elite	10
Probabilidad de Cruce	0.3
Probabilidad de Mutación	0.1
Número Máximo de Iteraciones	500

Tabla 4.8: Configuración del experimento a nivel de paquete

Se ha trabajado con la base de reglas que genera los patrones. La herramienta que ha automatizado las pruebas es similar a la empleada para el módulo de clasificación a nivel de paquetes y se basa en contrastar los resultados producidos por la base de reglas para cada conjunto de clasificaciones de paquetes, con las salidas producidas

por el módulo de correlación de alertas. Se han llevado a cabo 50.000 pruebas y los resultados coinciden con los expuestos en la tabla 4.9.

Categoría de aciertos	Porcentaje
Aciertos como etiqueta más probable	87,12 %
Aciertos como segunda etiqueta más probable	12,43 %
Aciertos como tercera etiqueta más probable	0,45 %
Total de aciertos	100 %

Tabla 4.9: Resultados del experimento a nivel de paquete

Como puede verse en la tabla 4.9, en todas las pruebas realizadas la solución propuesta por la base de reglas ha figurado en alguna de las posibles clasificaciones generadas por el algoritmo genético. En el 87,12 % se ha tratado de la clasificación marcada como la más probable, en el 12,43 % como la segunda más probable y en el 0,45 % como la tercera más probable. Además de acertar con los resultados, el módulo de clasificación a nivel de traza ha propuesto clasificaciones alternativas que no ha considerado la base de reglas. Por lo tanto es posible afirmar que los resultados han sido satisfactibles.

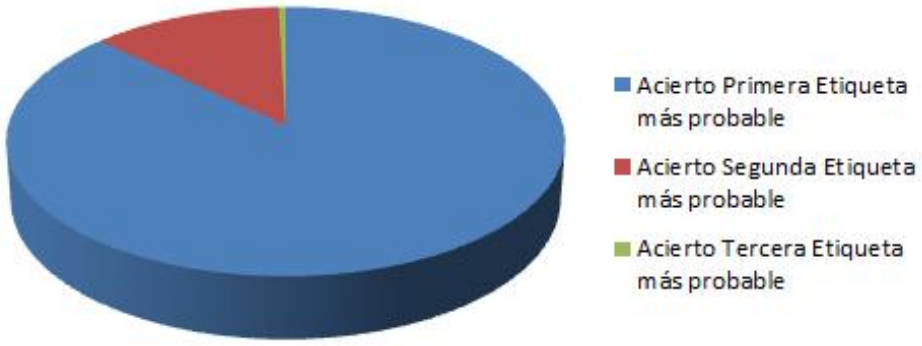

Figura 4.10: Resultados del experimento de clasificación a nivel de trazas

Capítulo 5

Resultados Experimentales

Para conocer hasta qué punto han sido alcanzados los objetivos que se han ido proponiendo a lo largo del presente trabajo, se han llevado a cabo diferentes experimentos en un escenario real. Para ello se ha dispuesto de las capturas de tráfico del año 2011 cedidas por el Centro de Cálculo de la Universidad Complutense de Madrid.

En este capítulo se explica detalladamente el escenario de las pruebas, y cada uno de los experimentos realizados tomando como referencia el origen del conjunto de tráfico que interviene en ella, y cada uno de los sistemas propuestos.

5.1. Escenario de pruebas para la evaluación

El escenario de pruebas para la evaluación consiste en la integración del NIDS APAP [83] con el sistema de correlación de alertas APACS.

En la figura 5.1 puede verse el esquema del escenario. A continuación se explica brevemente la configuración de cada uno de sus componentes.

5.1.1. Configuración de APAP

APAP (Advanced Payload Analyzer Preprocessor) es un Sistema de Detección de Intrusiones en Redes basado en la identificación de anomalías en la carga útil del tráfico de la red sobre la que opera, inspirado en la modificación de la estrategia de detección PAYL, denominada Anagram. Para ello lleva a cabo un modelado de los patrones de uso habitual y legítimo del medio, permitiendo reconocer indicios de la actividad que se desvía de ellos. Esta estrategia resulta muy eficaz contra amenazas desconocidas, y permite desenmascarar estrategias de evasión basadas en la ofuscación de *malware*. Comparando dicho modelo con el de varios ataques conocidos

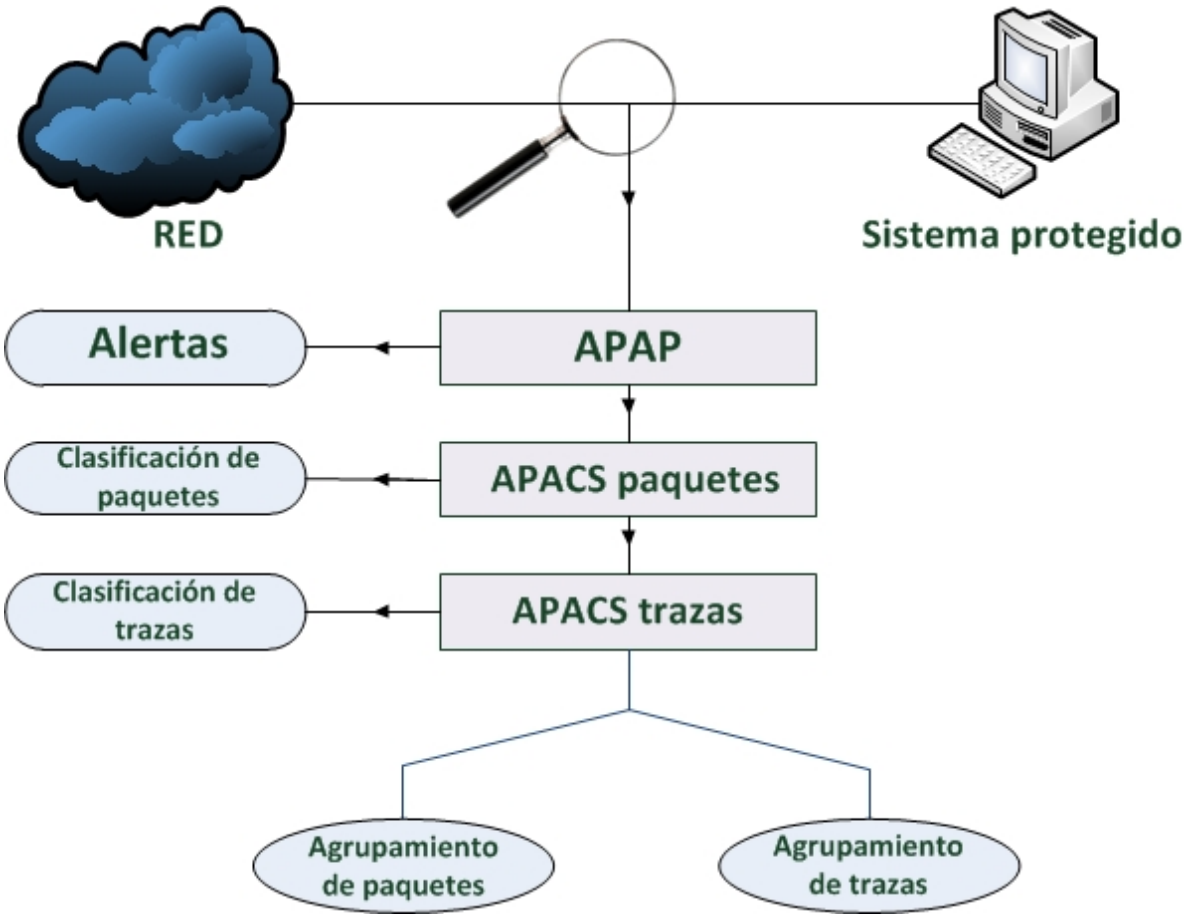

Figura 5.1: Escenario de pruebas de evaluación

se crea un conjunto de reglas que permiten detectar las anomalías presentes en el tráfico de la red a proteger.

Al analizar el tráfico, se lleva a cabo una comparación entre el modelo generado a partir del tráfico a etiquetar, y los valores de las reglas de detección establecidas en la fase de entrenamiento, correspondientes a actividades legítimas y maliciosas. De esta manera es posible determinar si el tráfico procesado es anómalo o corresponde al modo de uso habitual y legítimo de la red.

En la implementación, APAP ha sido configurado como un módulo preprocesador de otro NIDS, el conocido proyecto *open source* denominado Snort [84]. Snort ha sido diseñado para monitorizar el tráfico que circula por la red a la que brinda protección, empleando las librerías libpcap. El uso de dichas librerías proporciona una gestión sencilla y amigable de los paquetes que componen el tráfico de la red, pero tiene como contramedida, la tendencia a la generación de cuellos de botella al

auditar anchos de banda de más de 2 Gbps. Este problema está lejos de afectar al escenario de evaluación, ya que hasta la fecha, APAP no ha conseguido alcanzar la capacidad de análisis en tiempo real de 1 Gbps.

La decisión de integrar APAP en Snort parte de ventajosa arquitectura que provee el popular NIDS. Se trata de una arquitectura jerárquica similar a las que presenta el estándar ISO 18043 de NIDS [26].

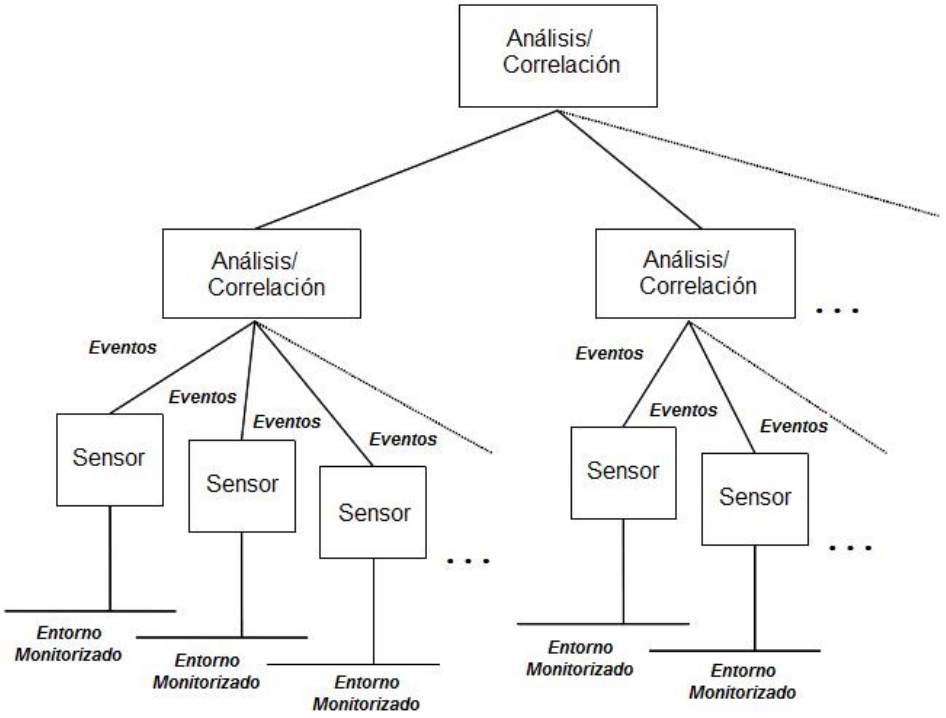

Figura 5.2: Arquitectura de Snort

La figura 5.2 muestra la arquitectura jerárquica de Snort. Como puede observarse, Snort se compone de un conjunto de módulos preprocesadores que analizan el tráfico de la red. Cada uno de ellos tiene una funcionalidad diferente: algunos simplemente preparan la información con el fin de evitar técnicas de evasión que exploten la falta de coherencia entre el sistema protegido y el NIDS. Por ejemplo los módulos Frag3 o Steam5 son parte de este grupo. Otros directamente se especializan en proteger un determinado servicio, como son el caso de HTTP Inspect, Pop o SMTP Preprocessor. Y por último están los módulos preprocesadores definidos por el usuario.

A través de una base de reglas, Snort sintetiza las alertas emitidas por cada uno de los módulos preprocesadores. Gracias a esto, APAP puede complementarse con el resto de preprocesadores, brindando una protección todavía mayor. No obstante, para el escenario de pruebas es interesante tratar las alertas de APAP por separado, por lo que han sido redirigidas a un fichero de alertas ajeno a Snort, y almacenadas acorde al estándar IDEMF [1] para el intercambio de alertas.

APAP precisa de cuatro fases de entrenamiento: inicialización, entrenamiento base, entrenamiento de referencia y entrenamiento de valores K_s definitivos. La fase de entrenamiento base y entrenamiento de referencias tienen como objetivo determinar aquellas características que representan de manera más significativa el modo de uso habitual y legítimo del sistema. Para ello se ha de disponer de un conjunto de trazas de tráfico legítimo.

Por otro lado, el entrenamiento en que se obtienen los valores K_s definitivos requiere de un conjunto de trazas de tráfico malicioso. En función de las características del experimento, el tráfico malicioso es dividido en diferentes conjuntos, que permitirán distinguir entre las reglas generadas para cada uno de ellos. Esas reglas son las encargadas de dirigir el etiquetado basado en criterios cualitativos llevado a cabo por APACS, y que dan pie a la evaluación del resto del sistema.

Como información adicional, es interesante mencionar que para recorrer la carga útil de los paquetes se ha elegido un tamaño de ventana N-gram de dimensión 3, acorde a las conclusiones presentadas en el trabajo de K. Riek et al. [85] y a los aclaraciones de K. Wang et al. [86] acerca del uso de las estructuras *Bloom Filter*.

5.1.2. Configuración de APACS

Una vez integrado APAP sobre Snort, se complementa mediante el sistema de correlación de alertas. De esta manera, APACS dispone en tiempo real de la información proveniente de las alertas emitidas por el NIDS, que resulta imprescindible para poder llevar a cabo su correcto etiquetado y un posterior agrupamiento. Como se explicó en el capítulo dedicado al sistema de correlación de alertas APACS, se cuenta con un módulo de análisis cuantitativo y un módulo de análisis cualitativo. Las técnicas con que han sido implementados en los experimentos son las mismas comentadas en los resultados parciales.

Módulo de análisis cuantitativo

El módulo de análisis cuantitativo ha sido integrado en las fases de entrenamiento de APAP. Para las pruebas en las que el conjunto de trazas de ataques son divididas, los valores centrales de los grupos son obtenidos mediante los datos procedentes de la unión de las distancias entre el tráfico legítimo y el tráfico malicioso, que han generado las reglas de cada uno de los grupos. De esta manera, el agrupamiento va a representar la unión de todos los grupos. El algoritmos de agrupamiento elegido ha sido K-means [87], y se ha forzado la generación de 6 grupos distintos.

Módulo de análisis cualitativo

El módulo de análisis cualitativo consta de una etapa de correlación a nivel de trazas y una etapa de correlación a nivel de paquetes. La etapa de correlación a nivel de paquetes ha sido implementada mediante una red neuronal artificial. Para ello se ha aprovechado la librería de C FANN [88]. Para cada experimento, dicha red neuronal ha sido entrenada acorde a las necesidades de los conjuntos de trazas de tráfico. Las características comunes vienen dadas por la tabla 5.1

Característica
Número de Entradas: T
Número de Salidas: 1
Número de Capas: 3 (1 oculta)
Porcentaje de error: 0.001
Número Máximo de etapas: 50000000
Número de Capas Ocultas: 1
Función de Activación: Elliot

Tabla 5.1: Configuración del módulo cualitativo de paquetes

La generación del conjunto de entrenamiento depende de la base del conocimiento propuesta. Dichas base del conocimiento ha sido elaborada acorde a la explicación del capítulo que presenta APACS, y dadas sus singularidades, serán definidas explícitamente al enunciar cada prueba de evaluación. Por otro lado, cabe destacar que la función de activación Elliot procede de las librerías FUNN, y se expresa como:

```
F.Elliot  span: 0 < y < 1
y = ((x*s) / 2) / (1 + |x*s|) + 0.5
d = s*1/(2*(1+|x*s|)*(1+|x*s|))
```

El módulo de análisis cualitativo a nivel de trazas aprovecha el etiquetado del módulo de clasificación de paquetes previamente descrito. Pero al igual que sucede a

nivel de paquete, es necesario conocer la base del conocimiento para poder definir los patrones asociados a cada una de sus reglas. Será definida al presentar las pruebas de evaluación.

La distancia Levenshtein [81] indica la aptitud de los individuos. Además de esto, la tabla 5.2 muestra el resto de parámetros del algorítmo genético.

Parámetro	Valor
Longitud del genotipo	6
Longitud de población inicial	200
Longitud de población de trabajo	50
Longitud de población elite	10
Probabilidad de cruce	0.3
Probabilidad de mutación	0.1
Número máximo de iteraciones	500

Tabla 5.2: Configuración del módulo cualitativo de trazas

Por ultimo, cabe mencionar que las operaciones de cruce y mutación también permanecen constantes en todas las pruebas. La operación de cruce consiste en establecer aleatoriamente una posición pivote común a los genotipos de los dos individuos que intervienen, e intercambiar sus extremos.

Por su parte, la operación de mutación recorre los distintos genes de un individuo. Para cada uno de ellos existe una probabilidad denominada probabilidad de mutación que decide si el gen permanece en su estado original o es sustituido por una característica al azar proveniente del conjunto a partir del cual se generaron los genes iniciales.

Otro aspecto importante a tener en cuenta, es el número de paquetes anómalos que van a ser considerados como una traza, ya que de ellos va a depender el conjunto inicial a partir del que se van a generar las poblaciones iniciales. Se ha elegido una cantidad de 20 paquetes por traza.

5.2. Evaluación mediante *datasets* de la UCM 2011

Debido a la poca precisión de la información disponible acerca del contenido de las trazas de tráfico DARPA99, el etiquetado emitido por el Centro de Cálculo de la Universidad Complutense de Madrid (UCM) para las trazas de tráfico analizado,

resulta esencial a la hora de medir la precisión de APACS. Para llevar a cabo el presente experimento, se ha distinguido entre el tráfico correspondiente al modo de uso habitual y legítimo de la subred de la UCM de la que proceden las capturas, y el tráfico anómalo correspondiente a los posibles intentos de intrusión. El Centro de Cálculo ha clasificado las anomalías detectadas durante el periodo de audición, de las siete maneras que se explican a continuación:

- **Actividad variada**
  (Misc Activity). Las anomalías pertenecientes a este grupo corresponden a actividades de características variadas que difieren del uso habitual y legítimo de la red. Las acciones que han desencadenado estas alertas proceden en su mayor parte de técnicas de enumeración, como solicitudes *echo request* (PING), consultas del estado de la red mediante *Traceroute*, así como intentos de acceso a puertos bloqueados.

  La carga útil de este tipo de paquetes incluye información adicional acerca del momento en el que se llevaron a cabo las solicitudes, y la información devuelta al extremo desde el que la consulta ha sido emitida. Su simplicidad incrementa el riesgo de que pase por tráfico legítimo. A pesar de ello, se trata de anomalías que salvo por algunas excepciones (técnicas de denegación de servicio) no suponen una amenaza directa, aunque pueden interpretarse como indicios de una fase de reconocimiento previa a un ataque.

- **Intentos potenciales de violación de privacidad corporativa**
  (Potencial Corporate Privacy Violation). Este agrupamiento de anomalías corresponde a alertas generadas por el uso de *software* potencialmente peligroso contra la privacidad corporativa. Normalmente se trata de aplicaciones habituales a nivel académico como Dropbox, Skype, MSN Messenger o herramientas que puede tener usos indebidamente peligrosos como IRC o Google Desktop. La carga útil de los paquetes involucrados en su utilización contiene huellas fácilmente detectables por los IDS. En ocasiones, son los propios desarrolladores quienes han dispuesto dichas huellas intencionadamente. Cuando se trata con información clasificada es importante considerar este tipo de amenazas.

- **Intentos de filtraciones de información**
  (Attempted Information Leak). Este agrupamiento contiene las anomalías originadas por intentos de enumeración del sistema. En las trazas de tráfico analizadas, esta categoría incluye frecuentemente intentos explotación de vulne-

rabilidades del navegador para la extracción de la información de las *cookies*. También es común la reiteración de acciones de inicio de sesión al sistema con credenciales incorrectos, o con el campo de contraseña incorrecto. A diferencia del agrupamiento de actividades variadas *(Misc Activity)*, suponen acciones con intenciones claramente ofensivas, fácilmente desenmascarables por el IDS.

- **Intentos de denegación de servicio**
  (Attempted Denial of Service). Los paquetes etiquetados como intentos de denegación forman parte de una reiteración de tráfico de características similares que pretende comprometer la disponibilidad del sistema. La mayor parte de este tráfico es muy difícil de detectarse únicamente con la supervisión de un NIDS que estudie la carga útil del sistema.

Para que APAP sea completamente eficaz contra esta amenaza, debe de considerar la información provista por los módulos preprocesadores de Snort encargados del procesamiento de los encabezados de los paquetes, y de esta manera llevar a cabo la construcción del escenario del ataque. Pero como previamente se indicó, para este experimento las alertas generadas por APAP son totalmente independientes de las del resto de módulos preprocesadores de Snort.

El tráfico seleccionado en esta categoría corresponde a las actividades detectables por el NIDS, como por ejemplo la reiteración de intentos de inicio de sesión con algún campo vacío. Otros ejemplos más claros son la presencia de paquetes que indican la presencia de determinadas estrategias dirigidas contra vulnerabilidades de los servidores DNS o vulnerabilidades *web*. A excepción de alguna circunstancia excepcional, los paquetes etiquetados en esta categoría constituyen acciones ofensivas premeditadas.

- **Intentos de adquisición de privilegios de usuario**
  (Attempted User Privilege Gain). Esta categoría abarca todos los paquetes de trazas de tráfico pertenecientes a intentos de escalada de privilegios, o de hacerse con el control del flujo de ejecución del sistema o alguna de sus aplicaciones. Este tipo de acciones son características del *malware*, y a excepción de los casos en que hayan sido ofuscados mediante técnicas de polimorfismo [89] [90], son fácilmente identificables.

Probablemente las estrategias más conocidas para hacerse con el control del flujo de ejecución de una aplicación son las conocidas técnicas de desborda-

miento de pila (*Stack Overflow*) [91] y desbordamiento de montículo (*Heap Overflow*) [92]. Ambas estrategias tienen en común la presencia de un trineo conocido como *Sled* de instrucciones, que tras explotar una vulnerabilidad del sistema arrastran el flujo de ejecución por sus direcciones de memoria hasta aquellas que contienen el *malware*. Un ejemplo típico de *Sled*, es el denominado *NOP Sled*, cuya característica principal es que todas sus instrucciones son nulas, conocidas como NOP en la mayor parte de arquitecturas actuales.

Además de estas técnicas, las trazas contienen otras más avanzadas, como por ejemplo, la pulverización de instrucciones sobre el montículo conocida como *Heap Spraying* [93]. Es evidente que la detección de un paquete con este tipo de contenido es una clara señal de la proximidad de *malware*, condicionando de esta manera, que se trate de uno de los grupos de mayor prioridad de tratamiento.

- **Presencia de código ejecutable**
  (Executabe Code was Detected). Esta categoría contiene paquetes con contenido ejecutable. La presencia de programas ejecutables no siempre va a representar una amenaza; de hecho, en la mayor parte de los casos no lo es. Pero es importante su etiquetado, ya que la aparición de paquetes con intentos de adquisición de privilegios en la misma traza, podría indicar que el programa se trate en realidad de *malware*. Además, determinadas políticas de restricción de privilegios a usuarios con niveles de acceso bajos, podrían servirse de su etiquetado para actuar con mayor precisión. En resumen, es conveniente que el NIDS detecte el código ejecutable que circula por la red a la que brida protección, con el fin de vigilar su comportamiento o restringir su acceso a usuarios poco privilegiados.

- **Presencia de troyano**
  (A Network Trojan was Detected). Los paquetes etiquetados como troyanos de red representan una amenaza directa contra el sistema, ya que explícitamente se está indicando que su contenido es *malware*. Este tipo de contenido es especialmente fácil de detectar para APAP, dado que su fase de diseño se ha centrado en detección de *malware*. Es la categoría con mayor prioridad de tratamiento.

La tabla 5.3 resume el porcentaje de aparición de paquetes etiquetados para cada una de las mencionadas categorías, en las trazas de tráfico cedidas por el Centro de

Cálculo de la Universidad Complutense de Madrid, auditadas en el año 2011.

Etiqueta	Porcentaje
Misc activity	33.8 %
Potential Corporate Privacy Violation	26.65 %
Attempted Information Leak	19.3 %
Attempted Denial of Service	16.2 %
Attempted User Privilege Gain	2 %
Executable Code was Detected	1.3 %
A Network Trojan was Detected	0.75 %

Tabla 5.3: Presencia de *malware* en *datasets* UCM 2011

El etiquetado predominante es el de las categorías de riesgo medio, como lo son los intentos de enumeración y los posibles ataques de denegación de servicio.

5.2.1. Entrenamiento de APAP

Antes de comenzar con los experimentos, APAP tiene que ser correctamente entrenado. Para ello, en primer lugar ha de construirse la estructura de datos *Bloom Filter* cuyo contenido represente el modo de uso habitual y legítimo de la red. En estas fases del entrenamiento participa el conjunto de tráfico limpio de los *datasets* previamente descritos.

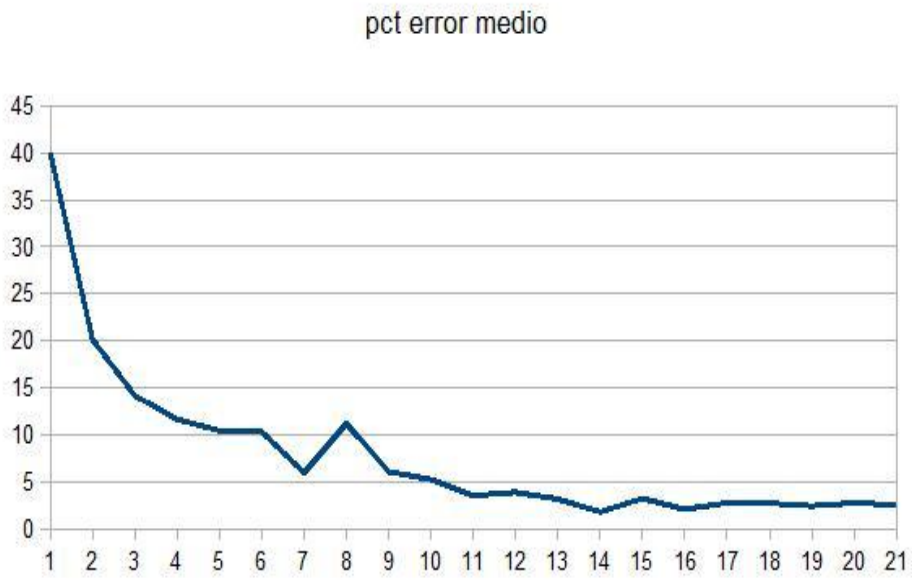

Figura 5.3: Porcentaje de error medio

La figura 5.3 muestra el porcentaje de error medio que indica la distancia entre la estructura *Bloom Filter* previa a un nuevo paso de entrenamiento, y la que se ha generado al realizarse el paso. El eje Y indica el porcentaje de error medio, y el eje X indica los distintos pasos de entrenamiento.

Para cada paso de entrenamiento se ha utilizado una selección de 300.000 paquetes del conjunto de entrenamiento, la cual ocupaba un espacio aproximado de 250 Mbytes. Es importante observar como el porcentaje de error medio tiende a la saturación, completándose el entrenamiento al alcanzar un valor aproximado del 2 %.

Llegado ese punto, la realización de nuevos pasos de entrenamiento no implica cambios representativos sobre el contenido del *Bloom Filter*, y por lo tanto resulta inútil continuar con el entrenamiento.

La figura 5.4 muestra la evolución del valor máximo contenido en la estructura *Bloom Filter* en los distintos pasos de entrenamiento. Dicho parámetro va a indicar el número de repeticiones del N-gram cuyo contenido resulta más característico. La importancia de este valor reside en las limitaciones computacionales del sistema sobre el que se ha desplegado APAP. Dado que la implementación de los módulos preprocesadores de Snort es en el lenguaje C, el valor máximo representable sin extender la estructura primitiva de los números enteros es 2.147.483.647. El valor

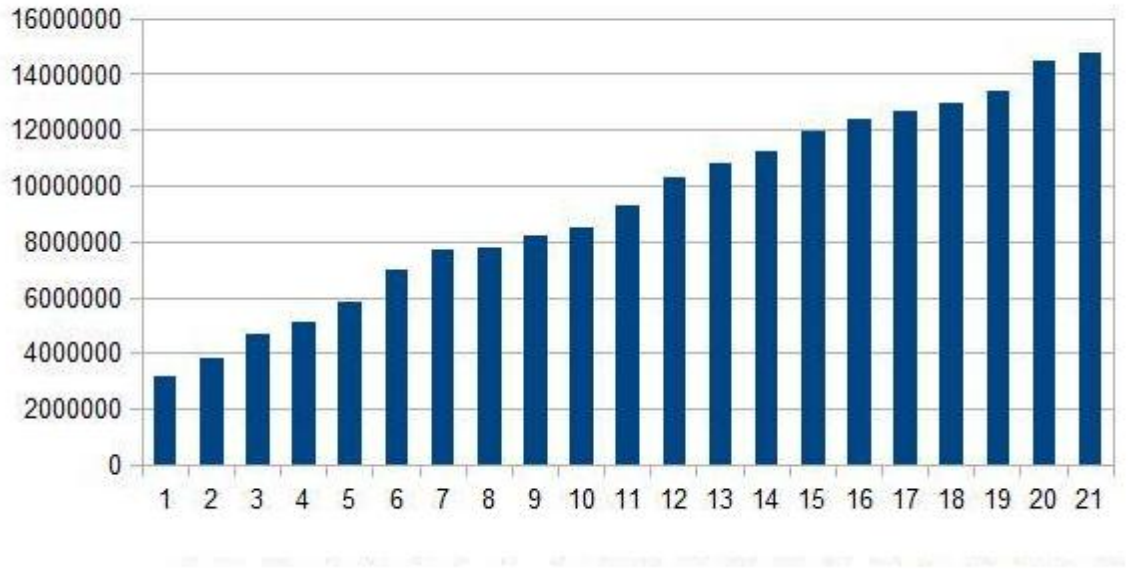

Figura 5.4: Máximo valor del *Bloom Filter*

máximo monitorizado ha sido 14.762.427. Esto es un indicador de que este parámetro no va a poder ser explotado mediante ataques de complejidad.

A continuación se lleva a cabo la fase de entrenamiento que genera las reglas de APAP. Esta etapa requiere del uso de parte de las colecciones de tráfico malicioso de los *datasets* de la UCM. La otra parte se reserva para realizar las pruebas de detección. Mediante distintos entrenamientos, es llevada a cabo la separación entre las reglas de APAP generadas para las diferentes categorías de anomalías. Dichas reglas son agrupadas en las reglas que APACS va a utilizar, y pueden verse en la tabla 5.4

Categoría	Reglas de APAP	Reglas de APACS
Misc activity	1..33	R1
Potential Corporate Privacy Violation	34..67	R2
Attempted Information Leak	68..93	R3
Attempted Denial of Service	94..108	R4
Attempted User Privilege Gain	109..115	R5
Executable Code was Detected	116..119	R6
A Network Trojan was Detected	120..122	R7

Tabla 5.4: Reglas de APACS en las pruebas de evaluación

5.2.2. Resultados de APAP

Antes de evaluar el comportamiento de APACS, es importante comprobar que APAP funciona correctamente. Si la etapa de detección de anomalías no se comporta de la manera esperada, llevar a cabo la correlación de sus alertas carece de sentido.

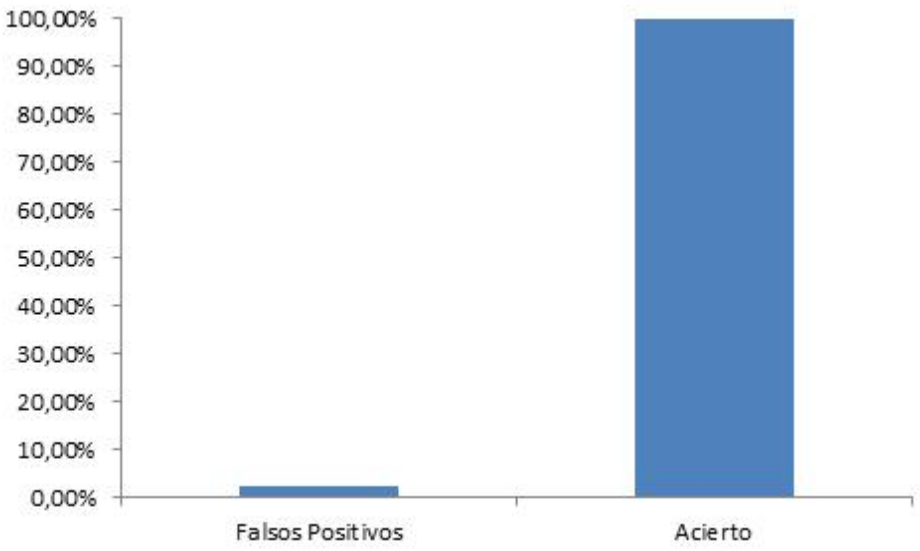

Figura 5.5: Resultados de APAP

La figura 5.5 muestra los resultados obtenidos por APAP al analizar el subconjunto de trazas de ataques reservado para las pruebas de detección, y el subconjunto de trazas del tráfico que representa el modo de uso habitual y legítimo del sistema, que no se utilizó en las dos primeras etapas del entrenamiento. La tasa de acierto ha sido del 100 %, pero la tasa de falsos positivos ha sido del 2.2 %. Este último valor no es demasiado deseable, ya que en redes con un modo de uso más homogéneo puede aproximarse mucho más al 0 %. El trabajo del sistema de correlación va a ser en parte mitigar este problema.

5.2.3. Resultados de APACS

Para interpretar los resultados de APACS, va a considerarse por separado el módulo de clasificación cuantitativa y el módulo de clasificación cualitativa.

Clasificación cuantitativa

Durante la etapa de entrenamiento de APAP en la que se definen los valores de las K_s definitivas, se ha generado un fichero con las distancias máximas normaliza-

das entre el espectro que representa el tráfico legítimo, y el espectro que representa el tráfico malicioso. Cuanto mayor distancia haya, mayor será la probabilidad de que la traza de tráfico analizada represente un verdadero ataque. Dicho fichero va a utilizarse como el conjunto de muestras a partir del cual se van a calcular los valores centrales de los grupos establecidos en el módulo de clasificación cuantitativa de APACS.

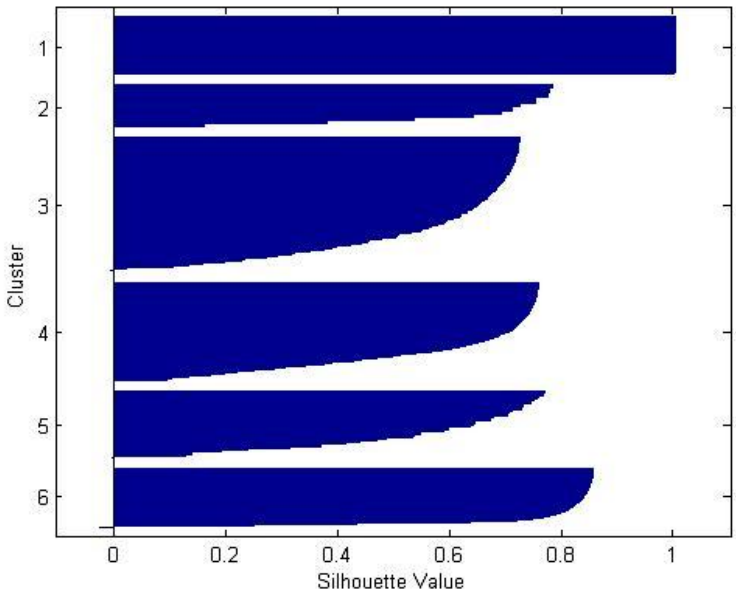

Figura 5.6: Pertenencia a cada agrupación

Tras aplicar el correspondiente algoritmo de agrupamiento, el diagrama de *silhouette* que indica el estado del agrupamiento corresponde con el de la figura 5.6. Es posible comprobar cómo existe una clara separación entre las muestras correspondientes a cada uno de los grupos.

Además, la tabla 5.5 indica el valor exacto de cada uno de los valores centrales, al que se le ha atribuido un grado de peligrosidad que oscila entre Muy Baja y Muy Alta. Aquellos grupos con valores más próximos a cero denotan los casos en que el tráfico analizado tiene mayor similitud con el tráfico legítimo, y por lo tanto implican menor riesgo. Sin embargo, los grupos cuyos valores centrales son más altos indican una mayor distancia, y por lo tanto una mayor probabilidad de amenaza Para medir la capacidad del módulo de clasificación cuantitativo de identificar los falsos positivos, se ha probado a analizar el subconjunto de tráfico que representa el modo de uso habitual y legítimo de las trazas de la UCM.

Agrupación	Valor Central	Riesgo
Grupo 1	0.004737	Muy Bajo
Grupo 2	0.966887	Muy Alto
Grupo 3	0.762419	Medio
Grupo 4	0,798229	Medio
Grupo 5	0,883234	Alto
Grupo 6	0,678284	Bajo

Tabla 5.5: Resultados de pertenencia con tráfico legítimo

La figura 5.7 y la tabla 5.6 muestran los resultados obtenidos en el análisis. Para la figura, el eje Y representa el porcentaje de aparición a un determinado grupo mientras que el eje X cada uno de los grupos. La tabla contiene los valores con los que se ha constituido la gráfica. Como puede verse, no hay tráfico clasificado en los grupos de riesgo, lo que es muy buena señal. Para una interpretación de los resultados con mayor claridad, se agrupan los dos grupos de riesgo Medio, y se ordenan los resultados en función del nivel de riesgo.

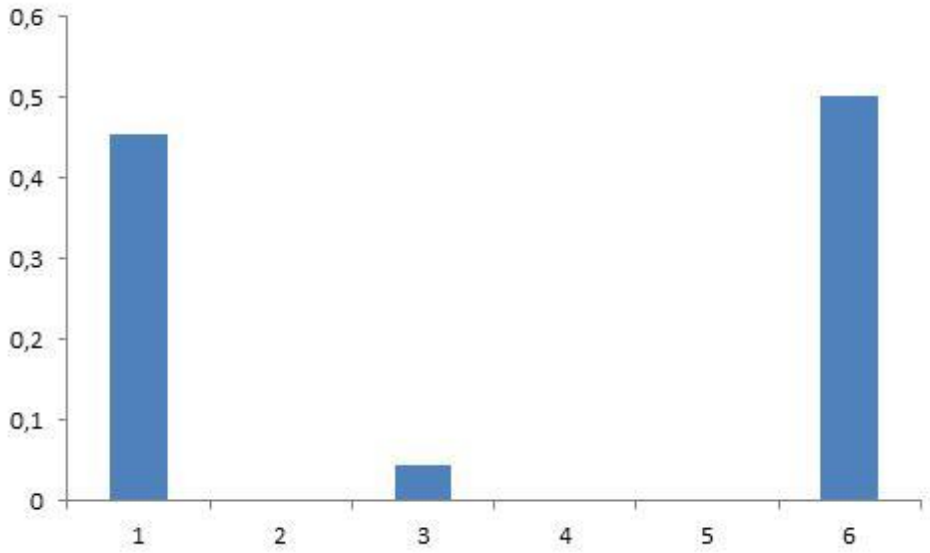

Figura 5.7: Resultados con tráfico legítimo

Agrupación	Porcentaje	Riesgo
Grupo 1	45.5 %	Muy Bajo
Grupo 2	0 %	Muy Alto
Grupo 3	4.3 %	Medio
Grupo 4	0 %	Medio
Grupo 5	0 %	Alto
Grupo 6	50.2 %	Bajo

Tabla 5.6: Resultados de riesgo con tráfico legítimo

La figura 5.8 y la tabla 5.7 muestran esquemas parecidos a los anteriores, pero considerando directamente el grado de riesgo.

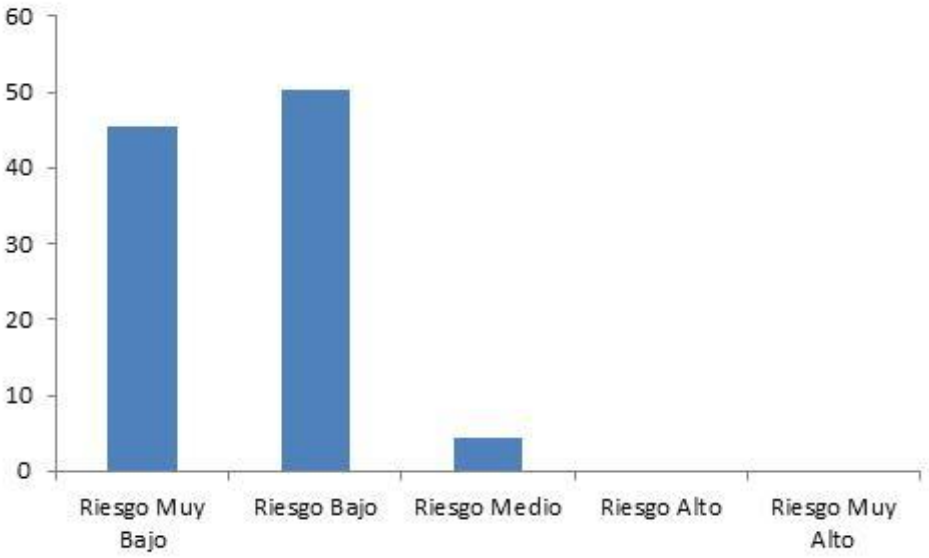

Figura 5.8: Resultados con tráfico legítimo

Agrupamiento	Porcentaje
Muy Bajo	45.5 %
Bajo	50.2 %
Medio	4.3 %
Alto	0 %
Muy Alto	0 %

Tabla 5.7: Resultados de experimento cuantitativo con ataques

El 95,7 % de los falsos positivos que ha generado el NIDS han sido catalogados como amenazas de riesgo Bajo o Muy Bajo, y el 4,3 % con riesgo medio. Es eviden-

te que el clasificador es capaz de identificar la mayor parte de los falsos positivos y etiquetarlos en niveles bajos, dando de esta manera prioridad al tratamiento de alertas verdaderas. Se puede concluir que se ha conseguido mitigar el efecto de los falsos positivos.

Pero antes de afirmar que el módulo se ha comportado de la manera esperada, es importante conocer el comportamiento en el caso de que las alertas correspondan con verdaderas amenazas. Para ello se ha analizado el subconjunto de trazas de tráfico correspondientes a amenazas, perteneciente al conjunto de trazas de la UCM.

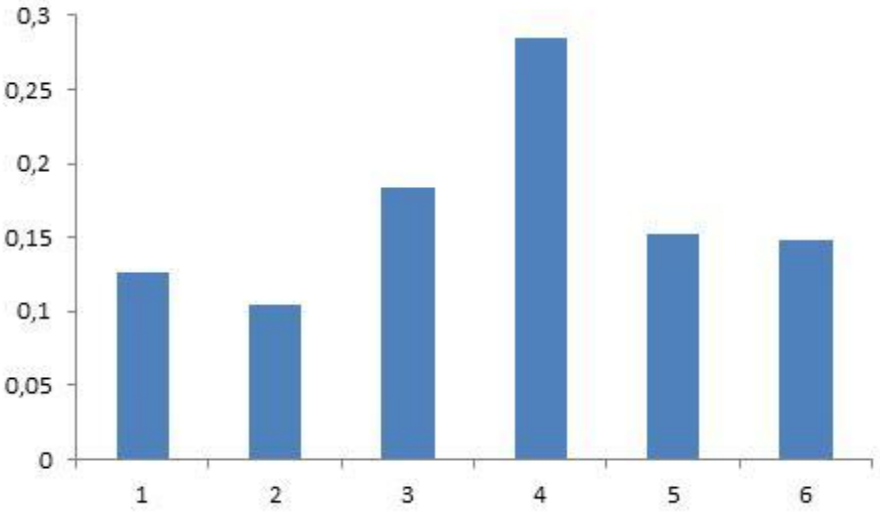

Figura 5.9: Resultados de ataques

Agrupación	Porcentaje	Riesgo
Grupo 1	12.6 %	Muy Bajo
Grupo 2	10.4 %	Muy Alto
Grupo 3	18.3 %	Medio
Grupo 4	28.5 %	Medio
Grupo 5	15.4 %	Alto
Grupo 6	14.8 %	Bajo

Tabla 5.8: Resultados experimentos cuantitativos de pertenencia de ataques

La figura 5.9 y la tabla 5.8 muestran el grado de pertenencia de los paquetes analizados a cada uno de los grupos. El eje X indica el grupo de pertenencia y el eje Y el porcentaje de alertas que han sido etiquetadas de una manera determinada. La tabla 5.8 muestra los porcentajes de pertenencia a cada grupo y el grado de riesgo

con el que ha sido catalogado. A modo de resumen se muestran los resultados para cada uno de estos grados de amenaza.

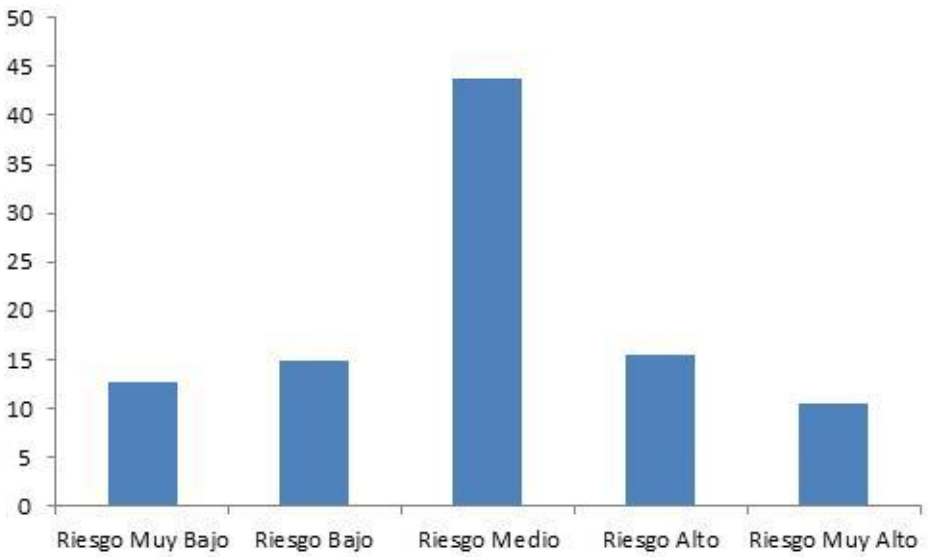

Figura 5.10: Resultados de ataques

Agrupamiento	Porcentaje
Muy Bajo	12.6 %
Bajo	14.8 %
Medio	43.8 %
Alto	15.4 %
Muy Alto	10.4 %

Tabla 5.9: Resumen de resultados de ataques cuantitativo

La figura 5.10 y la tabla 5.9 muestran el porcentaje de clasificaciones llevadas a cabo para cada factor de riesgo. Como puede verse, existe una tendencia *gaussiana* a la pertenencia a grupos de riesgo medio. Esto encaja bastante bien con los porcentajes de pertenencia a cada etiquetado en las trazas de tráfico cedidas por el Centro de Cálculo de la UCM.

A la vista de los resultados puede afirmarse que el módulo de clasificación cuantitativo ha cumplido los objetivos propuestos. Durante este experimento, ha sido capaz de identificar los falsos positivos y establecer un agrupamiento de las alertas de tráfico analizadas en función de que la probabilidad se trate de una verdadera amenaza o no.

Clasificación cualitativa a nivel de paquete

El módulo de clasificación cualitativa actúa en paralelo al módulo de clasificación cuantitativa. Si bien la clasificación cuantitativa agrupa las alertas en función de su probabilidad de tratarse de verdaderas amenazas, la clasificación cuantitativa a nivel de paquete tiene como meta el agrupamiento en función de la amenaza que contienen. Para ello va a requerir una base del conocimiento previa, similar a la explicada en el capítulo dedicado a APACS. Además tiene que ser capaz de reflejar la relación entre las distintas reglas activadas por APAPCS y su etiquetado.

N	Regla	Tag	Detalles
1	Si $Ord(0) = C_7$ Y $Ord(1, \cdots, n-1) = \emptyset$	E_1	Troyano
2	Si $Ord(0) = C_5$ Y $Ord(1, \cdots, n-1) = \emptyset$	E_2	Escalada de Privilegios
3	Si $Ord(0) = C_6$ Y $Ord(1, \cdots, n-1) = \emptyset$	E_3	Código Ejecutable
4	Si $Ord(0) = C_3$ Y $Ord(1, \cdots, n-1) = \emptyset$	E_4	Denegación de Servicio
5	Si $Ord(0) = C_2$ Y $Ord(1, \cdots, n-1) = \emptyset$	E_5	Software Provativo
6	Si $Ord(0) = C_4$ Y $Ord(1, \cdots, n-1) = \emptyset$	E_6	Enumeración I
7	Si $Ord(0) = C_1$ Y $Ord(1, \cdots, n-1) = \emptyset$	E_7	Enumeración II
8	Si $Ord(0) = C_7$ Y $Ord(1) = C_6$	E_8	Virus
9	Si $Ord(0) = C_7$ Y $Ord(1) = C_5$	E_8	Exploit
10	Si $Ord(0) = C_7$ Y $Ord(1) = C_2$	E_10	SW. Privativo SPY
11	Si $Ord(0) = C_7$ Y $Ord(1) = C_3$	E_11	Control Remoto
12	Si $Ord(0) = C_7$ Y $Ord(1) = C_4$	E_11	Control Remoto
13	Si $Ord(0) = C_5$ Y $Ord(1) = C_6$ Y $Ord(2) = C_2$	E_12	SW. Priv. no Auth
14	Si $Ord(0) = C_5$ Y $Ord(1) = C_6$	E_13	SW. Acceso no aut.
15	Si $Ord(0) = C_5$ Y $Ord(1) = C_3$ Y $Ord(2) = C_2$	E_12	SW. Priv. no Auth
16	Si $Ord(0) = C_5$ Y $Ord(1) = C_2$	E_12	SW. Priv. no Auth
17	Si $Ord(0) = C_6$ Y $Ord(1) = C_4$	E_14	Spyware I
18	Si $Ord(0) = C_6$ Y $Ord(1) = C_1$	E_15	Spyware II
19	Si $Ord(0) = C_3$ Y $Ord(1) = C_2$	E_10	SW. Privativo SPY
20	Si $Ord(0) = C_2$ Y $Ord(1) = C_4$	E_10	SW. Privativo SPY
21	Si $Ord(0) = C_2$ Y $Ord(1) = C_1$	E_10	SW. Privativo SPY
22	Si $Ord(0) = C_7$	E_1	Troyano
23	Si $Ord(0) = C_5$	E_2	Escalada de Privilegios
24	Si $Ord(0) = C_6$	E_3	Código Ejecutable
25	Si $Ord(0) = C_3$	E_4	Denegación de Servicio
26	Si $Ord(0) = C_2$	E_5	Software Provativo
27	Si $Ord(0) = C_4$	E_6	Enumeración I
28	Si $Ord(0) = C_1$	E_7	Enumeración II
29	*Otros*	E_16	Anomaía Desconocida

Tabla 5.10: Base del conocimiento en experimentos a nivel de paquete

La tabla 5.10 identifica las reglas que componen la base del conocimiento y el etiquetado emitido tras su activación. Las primeras reglas son más prioritarias que las últimas, activándose una sola regla por paquete. Para implementar la red neuronal artificial que dispara las etiquetas, se considera un total T de 7 reglas, una por cada regla de APACS. Los conjuntos de entrenamiento y la fase de entrenamiento se llevan a cabo como indica el capítulo de APACS. Al analizar el subconjunto de trazas de ataques cedido por la UCM, y reservado para las pruebas de detección, se obtienen los siguientes resultados.

Etiqueta	Amenaza	Porcentaje
E_1	Troyano	0,25357 %
E_2	Escalada de Privilegios	1,1332 %
E_3	Código Ejecutable	0,78212 %
E_4	Denegación de Servicio	16,2786 %
E_5	Software Privativo	8,458 %
E_6	Enumeración I	15,746 %
E_7	Enumeración II	28,792 %
E_8	Virus	0,01006 %
E_9	Exploit	0,0269 %
E_10	Software Privativo/SPY I	23,66832 %
E_11	Acción Control Remoto	0,42667 %
E_12	Software Privativo Acceso no autorizado	0,67145 %
E_13	Software Acceso no autorizado	0,0326 %
E_14	Spyware I	0,2725 %
E_15	Spyware II	0,581388 %
E_16	Anomalía Desconocida	0 %

Tabla 5.11: Resultados de experimentos cualitativos a nivel de paquete

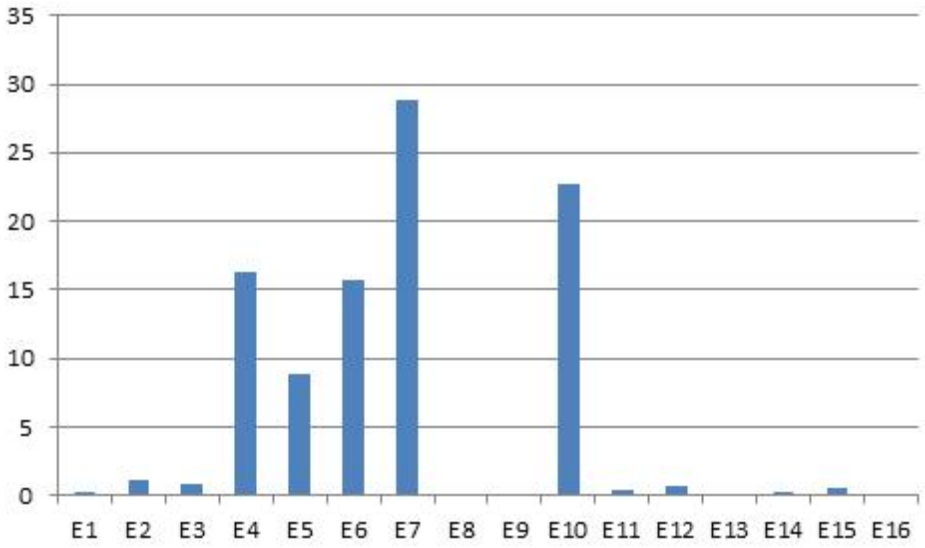

Figura 5.11: Porcentaje de pertenencia a cada grupo

La figura 5.11 y la tabla 5.11 muestran el porcentaje de pertenencia a cada grupo. En la figura el eje Y representa la etiqueta del grupo, y el eje X el porcentaje de alertas que contiene. A la vista de los resultados, la base del conocimiento propuesta ha sido capaz de agrupar las 53352 alertas emitidas por APAP en los 16 grupos establecidos. El módulo de clasificación cualitativo a nivel de paquete ha cumplido los objetivos propuestos.

Clasificación cualitativa a nivel de traza

Al igual que sucedió con la clasificación cualitativa a nivel de paquete, para llevar a cabo un estudio a nivel de trazas se requiere la construcción de una base del conocimiento. La finalidad de la base del conocimiento es indicar de qué manera se va producir la generación de los patrones asociados al genotipo de los individuos, tras cada paso de cruce o mutación.

N	Regla	Patrón	Tag	Detalles
1	Si $Ord(0) = E_1 4$ Y $Ord(1) = E_4$	$[14, 4, -, -]$	A_1	Botnet
2	Si $Ord(0) = E_4$ Y $Ord(1) = E_1 4$	$[4, 14, -, -]$	A_1	Botnet
3	Si $Ord(0) = E_2$ Y $Ord(1) = E_3$ Y $Ord(2) = E_8$	$[2, 3, 8, -]$	A_2	Ofuscado
4	Si $Ord(0) = E_3$ Y $Ord(1) = E_2$ Y $Ord(2) = E_8$	$[3, 2, 8, -]$	A_2	Ofuscado
5	Si $Ord(0) = E_2$ Y $Ord(1) = E_3$ Y $Ord(2) = E_1 3$	$[2, 3, 13, -]$	A_2	Ofuscado
6	Si $Ord(0) = E_3$ Y $Ord(1) = E_2$ Y $Ord(2) = E_1 3$	$[3, 2, 13, -]$	A_2	Ofuscado
7	Si $Ord(0) = E_8$ Y $Ord(1) = E_4$ Y $Ord(2) = E_2$	$[8, 4, 2, -]$	A_3	Gusano
8	Si $Ord(0) = E_8$ Y $Ord(1) = E_2$ Y $Ord(2) = E_4$	$[8, 2, 4, -]$	A_3	Gusano
9	Si $Ord(0) = E_1 0$ Y $Ord(1) = E_6$ Y $Ord(2) = E_1 2$	$[10, 6, 12, -]$	A_4	Drive-by
10	Si $Ord(0) = E_5$ Y $Ord(1) = E_4$	$[5, 4, -, -]$	A_4	Drive-by
11	Si $Ord(0) = E_1 2$ Y $Ord(1) = E_6$	$[12, 6, -, -]$	A_4	Drive-by
12	Si $Ord(0) = E_1 3$ Y $Ord(1) = E_6$	$[13, 6, -, -]$	A_4	Drive-by
13	Si $Ord(0) = E_1 2$ Y $Ord(1) = E_2$ Y $Ord(2) = E_1 4$	$[12, 2, 14, -]$	A_5	Adware
14	Si $Ord(0) = E_1 2$ Y $Ord(1) = E_2$ Y $Ord(2) = E_1 5$	$[12, 2, 15, -]$	A_5	Adware
15	Si $Ord(0) = E_1 3$ Y $Ord(1) = E_2$ Y $Ord(2) = E_1 4$	$[13, 2, 14, -]$	A_5	Adware
16	Si $Ord(0) = E_1 3$ Y $Ord(1) = E_2$ Y $Ord(2) = E_1 5$	$[13, 2, 15, -]$	A_5	Adware
17	Si $Ord(0) = E_1 4$	$[14, -, -, -]$	A_6	Spyware
18	Si $Ord(0) = E_1 5$	$[15, -, -, -]$	A_6	Spyware
19	Si $Ord(0) = E_1 0$	$[10, -, -, -]$	A_6	Spyware
20	Si $Ord(0) = E_6$	$[6, -, -, -]$	A_6	Spyware
21	Si $Ord(0) = E_8$	$[8, -, -, -]$	A_7	Virus
22	Si $Ord(0) = E_1$	$[1, -, -, -]$	A_8	Troyano
23	Otros	$[-, -, -, -]$	E_9	

Tabla 5.12: Base de reglas del experimento a nivel de traza

La tabla 5.12 muestra las reglas de la base del conocimiento, los patrones asociados, la etiqueta y los detalles sobre la actividad maliciosa detectada. El objetivo de las reglas es identificar la actividad característica del *malware* especificado en los detalles.

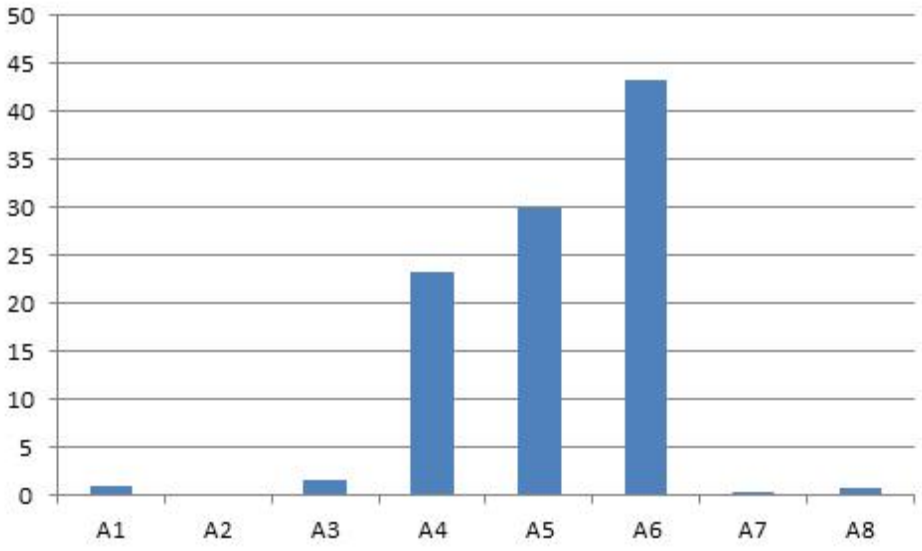

Figura 5.12: Porcentaje de etiquetado de anomalías a nivel de trazas

Etiqueta	Amenaza	Porcentale	Precisión
A_1	Botnet	0,91 %	89,1 %
A_2	Malware Ofuscado	0,141 %	73,1 %
A_3	Gusano	1,6 %	94,4 %
A_4	Drive-by	23,18 %	83,7 %
A_5	Adware	30,003 %	91,2 %
A_6	Spyware	43,2 %	71,3 %
A_7	Virus	0,236 %	83,1 %
A_8	Troyano	0,73 %	94,5 %
A_9	Otros	0 %	100 %

Tabla 5.13: Resultados del experimento a nivel de traza

La tabla 5.13 indica los resultados obtenidos tras el análisis de las trazas de anomalías de la UCM. En ella se indica la etiqueta, los detalles de la amenaza, el porcentaje del malware agrupado en la categoría y el porcentaje promedio aproximado que denota la precisión del etiquetado. En el capítulo que se presenta APACS, se explicó como la clasificación a nivel de trazas puede generar diferentes soluciones, y la probabilidad de que cada una sea la correcta. La tabla muestra las clasificaciones más probables, y el promedio de que esa clasificación sea la correcta.

La figura 5.12 representa el porcentaje de las alertas agrupado en cada categoría. El eje Y indica el porcentaje y el eje X cada una de las etiquetas. Por su parte, la figura 5.13 indica la precisión obtenida por la opción más representativa del algorit-

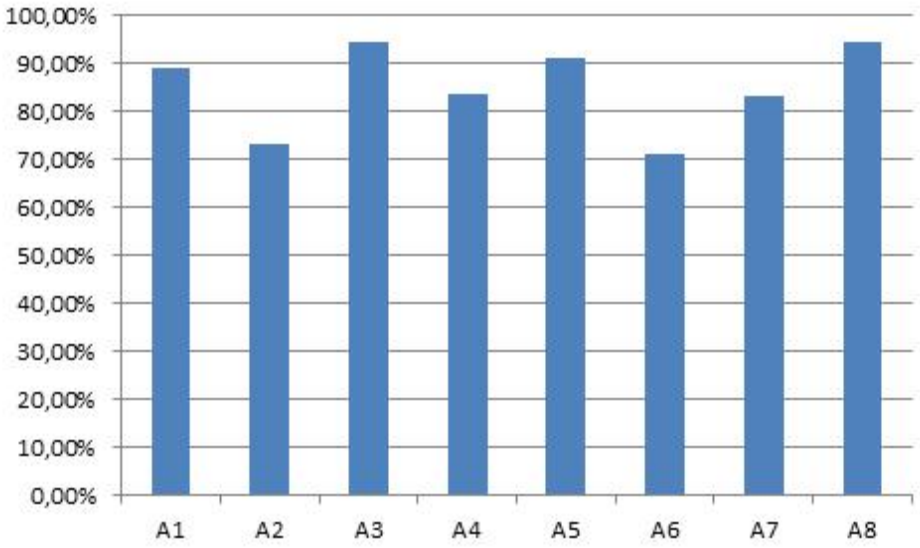

Figura 5.13: Precisión de la primera opción de etiquetado

mo genético de etiquetado. A la vista de los resultados, el sistema ha sido de capaz de clasificar las trazas de tráfico malicioso en las categorías propuestas, acorde a la base del conocimiento. De esta manera ha cumplido satisfactoriamente los objetivos propuestos.

5.3. Pruebas de precisión

Dado que el etiquetado propuesto por el Centro de Cálculo de la UCM corresponde a siete amenazas genéricas, es completamente imposible medir la precisión del sistema empleándolo como referencia. Esto es debido a que la base del conocimiento aplicada emite una clasificación mucho más detallada que la original.

El mismo problema sucede al tratar de evaluar la precisión obtenida sobre los conjuntos de trazas propuestos por DARPA, ya sea en el año 1999 o en el 2000. En algunos de los trabajos sobre sistemas de correlación basados en la construcción de escenarios de ataques, se ha utilizado un subconjunto de *datasets* publicados por DARPA especializado en ataques de denegación de servicio distribuido, siendo en ese caso una propuesta válida. Pero para evaluar el etiquetado de la carga útil, el problema de la falta de información detallada es persistente.

La solución a este problema consiste en evaluar los módulos a partir de datos generados de una manera similar a la utilizada para entrenar la red neuronal del

sistema. Es decir, se genera aleatoriamente un conjunto de muestras, cuyo contenido corresponde con la frecuencia de aparición de cada característica presentada en la base del conocimiento. Gracias a la base de reglas generada, es posible llevar a cabo un etiquetado con un 100 % de precisión. El proceso de evaluación consistió en contrastar muestra a muestra, el etiquetado propuesto por cada módulo de APACS, con el que ha propuesto la base de reglas.

En el caso del módulo de análisis a nivel cuantitativo, el problema es un poco más complejo. No es recomendable generar muestras aleatoriamente, ya que las amenazas deben seguir tratándose de anomalías. La prueba de precisión se ha llevado a cabo clasificando las trazas de ataques de la UCM en función de la probabilidad de amenaza. La prueba de evaluación consistió en contrastar el etiquetado manual de las trazas con el propuesto por APACS. Para llevar a cabo el etiquetado manual, se ha considerado como riesgo bajo el etiquetado del Centro de Cálculo de la UCM de la categoría de "Actividad variada". De riesgo medio el de las categorías de "Intentos potenciales de violación de privacidad corporativa" e "Intentos de filtraciones de información". El resto de categorías han sido consideradas como riesgo alto.

A continuación se detallan los resultados obtenidos en cada módulo

5.3.1. Precisión del módulo de análisis cuantitativo

El módulo de análisis cuantitativo ha sido probado con 85.000 paquetes, de los cuales 71.255 han sido etiquetados con éxito y 13.745 erróneamente. La tabla 5.14 muestra el porcentaje de acierto.

Tasa	Porcentaje
Acierto	83.82 %
Fallo	16.17 %

Tabla 5.14: Precisión del módulo de análisis cuantitativo

Se ha obtenido una tasa de acierto del 83.82 %, lo cual es bastante satisfactible. Este resultado depende directamente de la calidad del entrenamiento del NIDS, que lo que determina la distancia entre las características del tráfico legítimo y el paquete analizado. Posiblemente un mejor ajuste de APAP arroje mejores resultados.

5.3.2. Precisión del módulo de análisis a cualitativo a nivel de paquetes

La red neuronal del módulo de análisis cualitativo a nivel de paquetes ha sido probada con 50.000 muestras generadas aleatoriamente, y etiquetadas con la base del conocimiento. 49.756 muestras han sido etiquetadas con éxito y 244 erróneamente. La tabla 5.15 muestra el porcentaje de acierto.

Tasa	Porcentaje
Acierto	99.512 %
Fallo	0.488 %

Tabla 5.15: Precisión del módulo de análisis a cualitativo a nivel de paquetes

Se ha obtenido una tasa de acierto del 99.512 %, lo cual implica una precisión muy alta. A la vista del resultado, es evidente que el proceso de entrenamiento de la red neuronal artificial alcanzó el máximo número de iteraciones antes del error menor al 0.001 (después de varios días de entrenamiento). De cara a futuras implementaciones, es recomendable llevar el proceso de entrenamiento hasta el error deseado.

5.3.3. Precisión del módulo de análisis a cualitativo a nivel de trazas

Al igual que en el caso de la red neuronal, se ha medido la precisión del módulo de análisis cualitativo a nivel de trazas mediante un contraste con 50.000 muestras. Se ha conseguido clasificar con éxito el 100 % de las muestras. La tabla 5.16 muestra el porcentaje de acierto de cada una de las opciones planteadas.

Propuesta de solución	Porcentaje
Primera	75.124 %
Segunda	24.876 %

Tabla 5.16: Precisión del módulo de análisis a cualitativo a nivel de trazas

Como era de pensar, la mayor parte de los etiquetados correctos se concentra entre la primera y la segunda propuesta del algoritmo genético. Es evidente que los resultados obtenidos son excelentes.

5.4. Pruebas de rendimiento

Para medir la eficiencia del sistema, se ha aprovechado el sistema de evaluación del rendimiento integrado en APAP. Dicho sistema ha sido utilizado con anterioridad para comprobar el grado de mejora de diversas propuestas de optimización del algoritmo de detección [94] , aplicando técnicas de concurrencia.

Debido a que la medición del rendimiento en tiempo real es sensible al contexto del procesador y del sistema operativo, el tiempo de procesamiento se midió 20 veces, considerando el promedio. El equipo de pruebas dispone de un procesador Core 2 Duo, y la traza de tráfico examinada tiene un tamaño aproximado de 210 Kb, conteniendo únicamente paquetes maliciosos (es el peor de los casos). La tabla 5.17 muestra las penalizaciones de tiempo producidas en cada etapa de la correlación.

Fase del análisis	tiempo(milisegundos)
Análisis de la carga útil	171.104
Análisis cuantitativo	173.127
Análisis cualitativo (paquetes)	175.135
Análisis cualitativo (trazas)	221.362

Tabla 5.17: Pruebas de rendimiento con traza de 210Kb

Como es de esperar, el procesamiento de la carga útil del tráfico de la red implica el mayor esfuerzo computacional. El bloque propuesto para dar una respuesta en tiempo real se compone del módulo de análisis cualitativo y el módulo de análisis cuantitativo. Su sobrecarga en el sistema ha sido del 2.3 %, lo que muestra que su impacto es muy bajo.

Por otro lado, el modo de análisis a nivel de trazas ha causado una sobrecarga del 20.88 %. Es un valor relativamente alto, pero no afecta a los objetivos de la propuesta. Este módulo fue diseñado para llevar a cabo análisis forenses.

En resumen, APACS se ha comportado bien en lo referente al impacto sobre el rendimiento del sistema. Este experimento prueba que el rendimiento en el caso peor permite al bloque de procesamiento de paquetes en tiempo real, hacer su labor sin obstruir al NIDS.

Capítulo 6

Conclusiones, resultados y propuestas de trabajo futuro

6.1. Conclusiones

Se ha diseñado un sistema de correlación de alertas capaz de correlacionar la información proveniente de las alertas emitidas por el IDS en función de dos criterios muy diferentes. El primer criterio tiene un enfoque cuantitativo. Dado que el IDS que emite las alertas basa su comportamiento en la detección de anomalías del modo de uso habitual y legítimo del sistema, resulta interesante extraer información adicional que permita su etiquetado. Para ello se lleva a cabo un agrupamiento de las alertas considerando la distancia que separa el modo de uso habitual y legítimo del sistema, de la traza de tráfico que ha emitido cada alerta. En consecuencia es posible determinar a qué grupo de probabilidad de riesgo pertenece cada alerta, y condicionar su prioridad de tratamiento. Esta propuesta ha resultado especialmente eficaz para la detección de los falsos positivos, y por lo tanto para la mitigación de una de las amenazas más importantes de un NIDS: la denegación de servicio mediante la inyección de alertas poco prioritarias.

Por su parte, el procesamiento de alertas basado en un enfoque cuantitativo ha permitido determinar el grado de riesgo de las anomalías en función de una base del conocimiento. Para ello se lleva a cabo una clasificación a nivel de paquete y a nivel de traza. Los experimentos ha demostrados que ambas son capaces de agrupar con éxito las alertas. La primera de ellas permite extender la información que procede de las reglas que han activado la alerta, y la segunda, detectar ataques compuestos por secuencias de paquetes. Además, generan una visión global del contenido de cada traza de tráfico y permiten la extracción de información aprovechable para futuros

entrenamientos.

El sistema de correlación propuesto ha sido desplegado sobre un escenario de pruebas real, y ha cumplido satisfactoriamente cada uno de sus propósitos. Además, el impacto producido ha sido despreciable, evitando la penalización sobre la capacidad de procesamiento del NIDS en tiempo real.

6.2. Resultados

Durante la fase de elaboración, han sido aceptados o publicados los siguientes trabajos:

- **J. Maestre Vidal**, J. D. Mejía Castro, L. J. García Villalba. Q-APACS: Quantitative-based Anomaly Payload Alert Correlation System, VII Congreso Iberoamericano de Seguridad InformÁtica (CIBSI), Panamá, 2013 (Aceptado)
- **J. Maestre Vidal**, J. D. Mejía Castro, A. L. Sandoval Orozco, L. J. García Villalba. Evolutions of Evasion Techniques against network Intrusion Detection Systems. Proceedings of the 6th International Conference on Information Technology (ICIT), Amman, Jordania. Mayo 2013 (Publicado)
- **J. Maestre Vidal**, H. Villanúa Vega, J. D. Mejía Castro, L. J. García Villalba. Concurrency Optimization for NIDS (Poster Abstract). Proceedings of the 15th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID), Amsterdam, The Netherlands. Lecture Notes in Computer Science, Vol. 7462, pp. 395-396, September 2012. (Publicado)

6.3. Propuestas para futuros trabajos

Como posibles trabajos futuros pueden señalarse las propuestas que vienen a continuación.

Integración de herramientas de clasificación y agrupamiento alternativas

Aunque a lo largo del presente trabajo se han propuesto determinadas estrategias de clasificación y agrupamiento, existen muchas otras propuestas alternativas que podrían mejorar el rendimiento del sistema.

Agregación atributiva

Dado que el sistema de correlación de alertas propone diferentes clasificaciones en base a diferentes criterios, es interesante la incorporación de un sistema de agregación que considere clasificar las alertas en función del vector formado por cada uno de los distintos etiquetados emitidos.

Escenarios de ataques

Para la detección de ataques distribuidos, como por ejemplo la denegación de servicio distribuida, debe incorporarse al sistema un mecanismo que permita la construcción del escenario de cada ataque.

Estudio del comportamiento

El sistema de correlación propuesto ha sido diseñado para complementar NIDS basados en anomalías. Una interesante labor es la generalización del mismo hacia diferentes categorías de IDS, como por ejemplo los HIDS.

Bibliografía

[1] H. Debar, D. Curry, B. Feinstein. The Intrusion Detection Message Exchange Format (IDMEF). IETF RFC 4765, March 2007.

[2] H. Debar, A. Wespi. Aggregation and Correlation of Intrusion-Detection Alerts. In Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID), Davis, CA, USA. Lecture Notes in Computer Science Vol. 2212, pp. 85-103, October 2001.

[3] A. Valdes, K. Skinner. Probabilistic alert correlation. Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID), Davis, CA, USA. Lecture Notes in Computer Science, Vol. 2212, pp. 54-68, October 2001.

[4] A. Valdes, K. Skinner. Adaptive, Model-Based Monitoring for Cyber Attack Detection. In Proceedings of the third International Symposium on Recent Advances in Intrusion Detection (RAID), Toulouse, France. Lecture Notes in Computer Science Vol. 1907, pp. 80-93, October 2000.

[5] D. Andersson, M. Fong, A. Valdes. Heterogeneous Sensor Correlation: A Case Study of Live Traffic Analysis. In Proceedings of the IEEE Workshop on Information Assurance, United States Military Academy, West Point, NY, USA, pp. 30-37, June 2002.

[6] P. Porras, M. Fong, A. Valdes. A Mission Impact-Based Approach to INFOSEC Alarm Correlation. In Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID), Zurich, Switzerland. Lecture Notes in Computer Science Vol. 2516, pp. 95-114, October 2002.

[7] O. M. Dain, R. K. Cunningham. Building Scenarios from a Heterogeneous Alert Stream. In Proceedings of the IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, USA, pp. 1-6, June 2001.

[8] P. Ning, D. Xu. Adapting query optimization techniques for efficient intrusion alert correlation, Tech-nical report, NCSU, Department of Computer Science, 2002.

[9] F. Cuppens, F. Autrel, A. Miege, S. Benferhat, Recognizing malicious intention in an intrusion detection process. In Proceedings of the second International Conference on Hybrid Intelligent Systems, Santiago, Chile. Vol. 87, pp. 806-817, December 2002.

[10] F. Cuppens, A. Miege. Alert correlation in a cooperative intrusion detection framework. In Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, California, USA, pp. 202-215, May 2002.

[11] P. Ning, D. Xu. Learning attack strategies from intrusion alerts. In Proceedings of the 10th ACM Conference on Computer and Communications Security, Washington, DC, USA, pp. 200-209, October 2003.

[12] S. Noel, E. Robertson, S. Jajodia. Correlating intrusion events and building attack scenarios through attack graph distances. In Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC), Tucson, Arizona, USA, pp. 350-359, December 2004.

[13] B. Zhu, A. A. Ghorbani. Alert correlation for extracting attack strategies. International Journal of Net-work Security Vol. 3(3), pp. 244-258, 2006.

[14] R. Sadoddin, A. A. Ghorbani. An incremental frequent structure mining framework for real-time alert correlation. Computers Security Vol. 28(3). pp. 153-173, June 2009.

[15] H. T. Elshoush, I. M. Osman. Reducing false positives through fuzzy alert correlation in collaborative intelligent intrusion detection systems-A review. In Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ), Barcelona, Spain, pp. 1-8, July 2010.

[16] K. Alsubhi, I. Aib, R. Boutaba. FuzMet: a fuzzy logic based alert prioritization engine for intrusion detection systems. International Journal of Network Management, Vol. 22(4), pp. 263-284, July 2012.

[17] G. C. Tjhai, S. M. Furnell, M. Papadaki, N. L. Clarke. A preliminary two-stage alarm correlation and filtering system using SOM neural network and K means algorithm. Computers and Security, Vol. 29(6), pp. 712-723, September 2010.

[18] S. Peddabachigaria, A. Abrahamb, C. Grosanc, J. Thomasa. Modeling intrusion detection system using hybrid intelligent systems. Journal of Network and Computer Applications Vol. 30(1), pp. 114 - 132, January 2007.

[19] H. A. Nguyen, D. Choi. Application of Data Mining to Network Intrusion Detection: Classifier Selection Model. In Proceedings of the 11th Asia - Pacific Symposium on Network Operations and Management (APNOMS): Challenges for Next Generation Network Operations and Service Management, Beijin, China, pp. 399-408, October 2008.

[20] G. V. S. N. R. V. Prasad, Y. Dhanalakshmi, V. V. Kumar, I. Ramesh Babu. Modeling An Intrusion Detection System Using Data Mining And Genetic Algorithms Based On Fuzzy Logic. International Journal of Computer Science and Network Security(IJCSNS), Vol. 8(7), pp. 319-325, July 2008.

[21] Z. Bankovic, J. M. Moya, A. Araujo, S. Bojanic, O. Nieto-Taladriz. A Genetic Algorithm - based Solution for Intrusion Detection. Journal of Information Assurance and Security Vol. 4(3), pp. 192-199, 2009.

[22] G. E. Yujia, X. Feng, A. Zhang. Distributed Intrusion Detection System Using Autonomous Agents Based on DCOM. In Proceedings of the International Conference on Computer, Informatics, Cybernetics and Applications (CICA), Hangzhou, China. Lecture Notes in Electrical Engineering Vol. 107, pp. 1625-1631 2012.

[23] H. T. Elshoush, I. M. Osman. Intrusion Alert Correlation Framework: An Innovative Approach. In Pro-ceedings of the IAENG Transactions on Engineering Technologies, Special Volume of the World Congress on Engineering, London, UK. pp. 405-420, July 2012. Lecture Notes in Electrical Engineering Vol. 229, pp. 405-420, 2013.

[24] S. Roschke, F. Cheng, C. Meinel. A Flexible and Efficient Alert Correlation Platform for Distributed IDS. In Proceedings of the 4th IEEE International Conference on Network and System Security (NSS), Mel-bourne, Australia, pp. 24-31, September 2010.

[25] P. Porras, D. Schnackenberg, S. Staniford-Chen, M. Stillman, F. Wu. The common intrusion detection framework architecture. CIDF Working Group, resource available at http://gost.isi.edu/cidf/, 1999.

[26] ISO/IEC 18043:2006. Selection, deployment and operations of intrusion detection systems, 2006.

[27] R. Feiertag, C. Kahn, P. Porras, D. Schnackenberg, S. Staniford-Chen. A Common Intrusion Specification Language (CISL), resource available at http://gost.isi.edu/cidf/,1999.

[28] J. M. Estévez, P. García-Teodoro, J. E. Díaz-Verdejo. Anomaly detection methods in wired networks: a survey and taxonomy. Computer Communications Vol.27(16), pp. 1569-1584, October 2004.

[29] R. K. Cunningham, R. P. Lippmann, D. J. Fried, S. L. Garfinkel, I. Graf, K. R. Kendall, S. E. Webster, D. Wyschogrod, M. A. Zissman. Evaluating Intrusion Detection Systems without Attacking your Friends: The 1998 DARPA Intrusion Detection Evaluation. In Proceedings of the Third Conference and Workshop on Intrusion Detection and Response, San Diego, Cal, USA, 1999.

[30] R. P. Lippmann, R. K.Cunningham. Improving Intrusion Detection Performance using Keyword Selection and Neural Networks. In Proceedings of the Second International Symposium on Recent Advances in Intrusion Detection (RAID),West Lafayette, In, USA, September 1999. Lecture Notes in Computer Networks, Vol. 34(4), pp. 597-603, October 2000.

[31] J. McHugh. Testing intrusion detection systems: a critique to the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM transactions on Information and system Security, Vol.3(4), pp. 262-294, November 1999.

[32] M. Rash, A. Orebaugh, G. Clark, B. Pinkard, J. Babbin. Intrusion Prevention and Active Response: Deploying Network and host IPS. MA: Syngress, 2005.

[33] NSA. Security-Enhanced Linux. resource available at http://www.nsa.gov/research/selinux/index.shtml, Jan 2009.

[34] B. Morin, L. Me, H. Debar, M. Ducasse. M2D2: A Formal Data Model for IDS Alert Correlation. In Proceedings of the Fifth International Symposium on Recent Advances in Intrusion Detection (RAID), Zurich, Switzerland, October 2002. Lecture Notes in Computer Science, Vol. 2516, pp. 115-137, October 2002.

[35] W. Lee, S. J. Stolfo, K. W. Mok, A data mining framework for building intrusion detection models. In Proceeding of the 1999 IEEE Symposium on Security and Privacy, Oakland, California, USA, pp. 120-132, May 1999.

[36] U. Lindqvist, P. Porras. Detecting computer and network misuse through the production-based expert system toolset (P-BEST). In Proceedings of the IEEE

Symposium on Security and Privacy, Oakland, CA, USA. pp 146-161, May 1999.

[37] P. G. Neumann, P. A. Porras. EMERALD: Event monitoring enabling responses to anomalous live disturbances. Proceedings of the 20th NIST National Information Systems Security Conference, Baltimore, USA, pp. 353-365, October 1997.

[38] M. Roesch. Snort-lightweight intrusion detection for networks. In Proceedings of the 13th USENIX conference on System administration (LISA99), Seattle, WA, USA, pp. 229-238, November 1999.

[39] U. Zurutuza. Estado del arte Sistemas de Detección de intrusos. resource available at http://www.criptored.upm.es/guiateoria/gt_m399a.htm, 2004.

[40] S. L. Scott, P. Smyth. The Markov modulated Poisson process and Markov Poisson cascade with applications to web traffic data. In Proceedings of the Seventh Valencia International Meeting on Bayesian Statistics, Valencia, Spain, June 2002. Bayesian Statistics Vol. 7, pp. 671-680, 2003.

[41] D. Barbará, J. Couto, S. Jajodia, N. Wu. ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection. ACM SIGMOD Record, Vol. 30(4), pp. 15-24, December 2001.

[42] V. Jyothsna, V. V. Rama Prasad, K. Munivara Prasad, A Review of Anomaly based Intrusion Detection Systems. International Journal of Computer Applications, New York, USA, Vol. 28(7), pp. 26-35. August 2011.

[43] H. O. Sang, S. L. Won, An anomaly intrusion detection method by clustering normal user behavior, Computers and Security, Elsevier B.V. Amsterdam, The Netherlands, Vol. 22(7), pp. 596-612, October 2003.

[44] W. Yu, X. Y. Zhou, An adaptive method for anomaly detection in symmetric network traffic, Computers and Security, Elsevier B.V. Amsterdam, The Netherlands, Vol. 26(6), pp. 427-433, September 2007.

[45] N. Ye, Y. Zhang, C. M. Borror, Robustness of the Markov-chain model for cyberattack detection, IEEE Transactions on Reliability, Vol. 53(1), pp. 116-123, March 2004.

[46] Ch. Chen, Y. Chen, H. Lin. An efficient network intrusion detection, Computer Communications, Elsevier B.V. Amsterdam, The Netherlands, Vol. 33(4), pp. 477-484, March 2010.

[47] Y. Jiang, Y. Zhong, S. Zhang, Distributed Intrusion Detection for Mobile Ad Hoc Networks. In Proceeding of the Symposium on Applications and the Internet, Trento, Italy, pp. 94-97, January-February 2005.

[48] T. Tran, I. Aib, E. Al-Shaer, R. Boutaba. An Evasive Attack on SNORT Flowbits. Proceeding of the IEEE Network Operations and Management Symposium (NOMS), Maui, HI, USA, pp. 351-358, April 2012.

[49] H. Song, J. W. Lockwood, Efficient packet classification for network intrusion detection using FPGA. In Proceedings of the 2005 ACM/SIGDA 13th International Symposium on Field-programmable Gate Arrays, Monterey, California, USA, pp. 238-245, February 2005.

[50] M. V. Mahoney, P. K. Chan. PHAD: packet header anomaly detection for identifying hostile network traffic, Florida Institute of Technology, Melbourne, USA, Technical Report, 2001.

[51] K. Wang, S. J. Stolfo. Anomalous Payload-based Network Intrusion Detection. In Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID), Sophia Antipolis, France. Lecture Notes in Computer Science, Vol. 3224, pp. 203-222, September 2004.

[52] D. Bolzoni, S. Etalle, P. Hartel, E. Zambon. POSEIDON: a 2-tier Anomaly-Based Network Intrusion Detection System. In Proceedings of the Fourth IEEE International Workshop on Information Assurance, London, United Kingdom, pp. 144-156, April 2006.

[53] M. Bishop. A taxonomy of (Unix) system and network vulnerabilities, Technical Report CSE-9510, Department of Computer Science, University of California at Davis, May 1995.

[54] T. Zang, X. Yun, Y. Zhang. A Survey of Alert Fusion Techniques for Security Incident. In Proceedings of the ninth IEEE International Conference on Web-Age Information Management (WAIM), Zhangjiajie, China, pp. 475-481, July 2008.

[55] D. L. Mills. Network Time Protocol Specification, Implementation and Analysis. IETF RFC 1305, March 1992.

[56] S. Staniford, J. A. Hoagland, J. M. McAlerney. Practical automated detection of stealthy portscans. Journal of Computer Security, 10(1), pp. 105-136, 2002.

[57] K. Julisch. Mining alarm clusters to improve alarm handling efficiency. In Proceedings of The 17th Annual Computer Security Applications Conference (ACSAC 2001), New Orleans, Louisiana, USA, pp. 12-21, December 2001.

[58] F. Cuppens, R. Ortalo. LAMBDA: A Language to Model a Database for Detection of Attacks. Proceedings of the third International Symposium on Recent Advances in Intrusion Detection (RAID), Toulouse, France. Lecture Notes in Computer Science Vol. 1907, pp. 197-216, 2000.

[59] H. T. Elshoush, I. M. Osman. Alert correlation in collaborative intelligent intrusion detection systems-A survey. Applied Soft Computing, Vol. 11(7), pp. 4349-4365, October 2001.

[60] F. Cuppens. Managing alerts in a multi-intrusion detection environment. In Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC), New Orleans, Lo, USA, pp. 22-31, December 2001.

[61] C. V. Zhou, C. Leckie, S. Karunasekera. A survey of coordinated attacks and collaborative intrusion detection. Computer Security, Vol. 29(1), pp. 124-140, February 2010.

[62] B. Morin, H. Debar. Correlation of intrusion symptoms: an application of chronicles. Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection (RAID), Pittsburgh, PA, USA. Lecture Notes in Computer Science, Vol. 2820, pp. 94-112, September 2003.

[63] P. Ning, Y. Cui, D. S. Reeves, D. Xu. Tools and techniques for analyzing intrusion alerts, ACM Transactions on Information and System Security (TISSEC), Vol. 7(2),pp. 273-318, May 2004.

[64] P. Ning, D. Xu. Hypothesizing and reasoning about attacks missed by intrusion detection systems. ACM Transactions on Information and System Security (TISSEC), Vol. 7(4), pp. 591-627, November 2004.

[65] B. Morin, L. Me, H. Debar, M. Ducasse. M2D2: a formal data model for IDS alert correlation. In Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID), Zurich, Switzerland. Lecture Notes in Computer Science, Vol. 2516, pp. 115-137, October 2002.

[66] D. Xu, P. Ning. Alert correlation through triggering events and common resources. In Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC), Tucson, AZ, USA, pp.360-369, December 2004.

[67] C. Giovanni: Fun with packets: Designing a stick. Technical Report, Endeavor Systems, 2001.

[68] G. Tedesco, U. Aickelin. Strategic Alert Throttling for Intrusion. Proceedings of the 4th International Conference on Information Security (WSEAS), Tenerife, Canary Islands, Spain, pp. 246-251, May 2005.

[69] B. Balcerek, B. Szurgot, M. Uchronski, W. Waga. ACARM-ng: Next Generation Correlation Framework. Lecture Notes in Computer Science, Vol. 7136, pp. 114-127, 2012.

[70] R. Hunt, J. Slay. A New Approach to Developing Attack Taxonomies for Network Security including Case Studies. In Proceedings of the 17th IEEE International Conference on Networks (ICON), Singapore, pp. 281-286, December 2011.

[71] S Shiva, C. Simmons, C. Ellis, C. Dasgupta, Q. Wu. Wu. AVOIDIT: A Cyber Attack taxonomy. Technical report. University of Memphis, August, 2009.

[72] N. Gruschka, J. Meiko. Attack surfaces: A Taxonomy For Attacks on Cloud Services. Proceedings of the 3rd IEEE International Confered on Cloud Computing (CLOUD), Miami, Florida, USA, pp. 276-279, July 2010.

[73] M. V. Mahoney, P. K. Chan. An analysis of the 1999 Darpa/Lincoln Laboratory Evaluation Data for Net-work Anomaly Detection. Proceedings of the 6th International Symposium of Research in Attacks, Intrusions, and Defenses (RAID), Pittsburgh, PA, USA. Lecture Notes in Computer Science, V. 2820, pp. 220-237, September 2003.

[74] M. J. Hashmi, S. Manish, S. Rajesh. Classification of DDoS Attacks and their Defense Techniques using Intrusion Prevention System. International Journal of Computer Science and Communication Networks, Vol. 2(5), pp. 607-614, November 2012.

[75] H. Berghel. Hiding data, forensics, and anti-forensics. Communications of the ACM Vol. 50(4), pp. 15-20, April 2007.

[76] T. H. Ptacek, T. N. Newsham. Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection. Technical Report, Secure Networks Inc Calgary Alberta, January 1998.

[77] A. Kamrani, R. Wang, R. Gonzalez. A Genetic Algorithm Methodology for Data Mining and Intelligent Knowledge Acquisition. Computers and Industrial Engineering, Vol. 40(4), pp. 361-377, September 2001.

[78] N. A. Barricelli. Numerical Testing of Evolution Theories. Acta Biotheoretica Vol. 16(1), pp. 69-98, 1962

[79] A. Fraser, D. Burnell. Computer Models in Genetics. New York: McGraw-Hill, 1970.

[80] I. Flockhart, N. J. Radcliffe. A Genetic Algorithm-Based Approach to Data Mining. Proceedings of the 2nd International Conference on Knowledge Discovery Data Mining (KDD), Portland, Oregon, USA, pp. 299-302, August 1996.

[81] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. Soviet Physics-Doklady, Vol. 10(8), pp. 707-710, February 1966.

[82] R. Hamming. Error Detecting and Error Correcting Codes. Bell System Technical Journal, Vol. 29(2), pp. 147-160, April 1950.

[83] L. J. Garcia Villalba, J. D. Mejia Castro, A. L. Sandoval Orozco, J. Martinez Puentes. Malware Detection System by Payload Analysis of Network Traffic (Poster Abstract). In Proceedings of the 15th International Symposium of Research in Attacks, Intrusions, and Defenses (RAID), Amsterdam, The Netherlands. Lecture Notes in Computer Science, Vol. 7462, pp. 397-398, September 2012.

[84] Snort: Open source network intrusion detection system. Available from: http://www.snort.org, 2002.

[85] K. Rieck, P. Laskov. Detecting Unknown Network Attacks Using Language Models. In Proceedings of the Third International Conference on Detection of Intrusions, Malware and Vulnerability Assessment(DIMVA), Berlin, Germany, July 2006. Lecture Notes in Computer Science Vol. 4064, pp. 74-90, July 2006.

[86] K. Wang, J. J. Parekh, S. J. Stolfo. Anagram: A Content Anomaly Detector Resistant to Mimicry Attack. Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID), Hamburg, Germany. Lecture Notes in Computer Science, Vol. 4219, pp. 226-248, September 2006.

[87] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation.

IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24(7), pp. 881-892, 2002.

[88] S. Nissen, E. Nemerson. Fast artificial neural network library. Resource available at http://leenissen.dk/fann/wp/

[89] P. O'Kane, S. Sezer, K. McLaughlin. Obfuscation: The Hidden Malware. IEEE Security and Privacy, Vol. 9, No. 5, pp. 41-47, October 2011.

[90] I. You, K. Yim. Malware Obfuscation Techniques: A Brief Survey. Proceedings of the 5th International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA), Fukuoka, Japan, pp. 297-300, November 2010.

[91] J. Regehr, A. Reid, K. Webb. Eliminating stack overflow by abstract interpretation. ACM Transactions on Embedded Computing Systems (TECS), Vol. 4(4), pp. 751-778, November 2005.

[92] Y. Wang, D. Gu, J. Xu, M. Wen, L. Deng. RICB: Integer overflow vulnerability dynamic analysis via buffer overflow. In Proceedings of the Third International ICST Conference on Forensics in Telecommunications, Information, and Multimedia, Shanghai, China. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 56, pp. 99-109, November 2011.

[93] M. Egele, P. Wurzinger, C. Kruegel, E. Kirda. Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks. In Proceedings of the 6th International Conference of Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), Como, Italy. Lecture Notes in Computer Science, Vol.5587, pp. 88-106, July 2009.

[94] J. Maestre Vidal, H. Villanúa Vega, J. D. Mejía Castro, L. J. García Villalba. Concurrency Optimization for NIDS (Poster Abstract). Proceedings of the 15th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID), Amsterdam, The Netherlands. Lecture Notes in Computer Science, Vol. 7462, pp. 395-396, September 2012.