Sistema de Detección de Intrusiones con Mantenimiento Asistido de Bases de Datos de Ataques Mediante Aprendizaje Automático

José Ignacio Fernández-Villamor
Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid
jifv@gsi.dit.upm.es

Mercedes Garijo
Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid
mga@dit.upm.es

***Resumen*—Los sistemas de detección de intrusiones (o IDS, del inglés *Intrusion Detection System*) tienen como fin la detección de ataques en redes de comunicaciones. Como tales, constituyen un elemento de interés en la provisión de seguridad en gestión de redes ante la asunción de existencia de agujeros de seguridad en los sistemas hardware y software. Por otro lado, existen sistemas de detección de intrusiones de código abierto basados en reglas, cuya principal desventaja consiste en el esfuerzo técnico de matenimiento de la base de datos de reglas. En este documento se analizan las técnicas más utilizadas en sistemas de detección de intrusiones y se reutilizan sistemas de intrusiones basados en reglas para proponer un sistema de detección de intrusiones con mantenimiento asistido de bases de datos de ataques mediante aprendizaje automático.**

I. Introducción

La gestión de red, considerando el modelo FCAPS (*Fault, Configuration, Accounting, Performance and Security*), involucra las ramas de gestión de incidencias, configuración, contabilidad, prestaciones y seguridad. Los sistemas de detección de intrusiones (o IDS, del inglés *Intrusion Detection Systems*) [1] se encuadran dentro de la gestión de seguridad y tienen como fin la detección de ataques mediante análisis de tráfico de red y otros datos en zonas desmilitarizadas. Como tales, constituyen un elemento de interés en la provisión de seguridad en redes ante la asunción de existencia de agujeros de seguridad en los sistemas hardware y software.

En este documento se analizan las técnicas más utilizadas en sistemas de detección de intrusiones y se propone una solución de este tipo que asiste en el mantenimiento y gestión de la base de datos de ataques.

II. Trabajos relacionados

Típicamente, los sistemas de detección de intrusiones son responsables de la identificación de patrones de comportamiento anómalos o susceptibles de serlo conforme a una determinada política de seguridad. Por ejemplo, una política de seguridad podría limitar los servicios accesibles a un conjunto de usuarios, al margen de reglas obvias como bloquear otro tipo de comportamientos como la introducción de código malicioso que pudiese provocar daños en equipos de red como servidores o terminales cliente. Así, en la definición de una política de seguridad de un sistema de detección de intrusiones típicamente se ha conocido la diferenciación entre *Anomaly Detection Systems* [2], que basan la detección en la identificación de comportamientos diferentes al típico de un usuario, y *Misuse Detection Systems* [3], que basan la detección en la identificación de comportamientos conocidos de ataques. En cualquier caso, de una política de seguridad se deduce todo el conjunto de actividades lícitas en una red, siendo la labor de un sistema de detección de intrusiones la notificación de un ataque a otro elemento del sistema o al propio administrador mediante alarmas. Alternativamente, planteados como la evolución de los sistemas de detección de intrusiones están los sistemas de prevención de intrusiones [4], [5], [6] o IPSs (*Intrusion Prevention Systems*), que extienden la responsabilidad de detección a la de evitar las intrusiones, considerándose alternativas como la integración en forma de intermediarios cortafuegos u otras como la distribución de las funciones de detección y actuación a diversos módulos o agentes, pudiendo asimilarse a un sistema de autogestión de incidencias regido por alarmas de seguridad.

Existen diversos enfoques en la implementación de los sistemas de detección de intrusiones. En primer lugar, un enfoque utilizado tradicionalmente consiste en, de forma análoga a los tradicionales antivirus personales, almacenar manualmente un conjunto de vulnerabilidades y sus mecanismos de detección e identificación para su aplicación en los datos de análisis [7]. Esta alternativa implica mantener actualizada una base de datos de vulnerabilidades junto con sus mecanismos de detección para asegurar la utilidad del sistema de detección. Un enfoque similar y propio de los ADSs consistiría en, suponiendo un entorno web, almacenar todas las rutas posibles de navegación con un rastreador para después bloquear aquéllas que no se correspondan con alguno de esos patrones. Igualmente, esta solución adolecería de problemas parecidos, como la imposibilidad de mantener una base de datos de este tipo incluso de forma automática al poder existir infinidad de patrones de navegación, que no pueden ser almacenados de forma explícita en una base de datos, por naturaleza, de

dimensión finita.

En segundo lugar, existe la alternativa de emplear técnicas de sistemas inteligentes [8] para la detección de intrusiones, ya sean historiales de datos tratados en segundo plano y con posterioridad [9], o datos de tráfico procesados en tiempo real [10]. Esto permite la utilización de heurísticos para detectar comportamientos anómalos y obviar una posible base de datos de vulnerabilidades, o bien actualizar la base de datos mediante aprendizaje automático aplicando técnicas como redes neuronales o mapas topológicos autoorganizativos (o SOM, de *Self-Organizing Maps*) [11].

Finalmente, existen arquitecturas multiagente [12] que utilizan estas técnicas para la implementación de sistemas de detección de intrusiones distribuidos, de forma que se mejoren las características de detección.

A continuación se detallan las técnicas utilizadas en proyectos relacionados de sistemas de detección de intrusiones con aprendizaje automático de datos de tráfico y código.

II-A. Clasificación automática

Una de las tareas en los sistemas de detección de intrusiones es la parte de detección e identificación de una intrusión a partir de los datos de entrada de que puede disponer el sistema. Actualmente, con técnicas de inteligencia artificial puede llevarse a cabo una clasificación automática de la información de entrada, que puede consistir en datos de tráfico de red o de código de procesos (como son el tipo de protocolo, servicio, número de octetos, número de fallos en el acceso a un sistema, número de creaciones de ficheros, tasas de error, etc.), en los diferentes ataques posibles tras un proceso de aprendizaje por parte del sistema.

Para la prueba, trabajo y refinamiento de dichas técnicas, suelen usarse conjuntos de datos de pruebas como el de *The Third International Knowledge Discovery and Data Mining Tools Competition* [13]. Este conjunto de datos está formado por casi cinco millones de conexiones de nivel de transporte clasificadas en diferentes ataques (adivinación de contraseña, vulnerabilidad de servidor web, ataque de desbordamiento de zonas de memoria, monitorización de puertos, etc.) o en tráfico normal, pudiendo agruparse, a su vez, esta clasificación de tráfico en diferentes grupos:

- *Normal*: Tráfico normal.
- *Probe*: Correspondiente a ataques de monitorización o introspección en un sistema para encontrar vulnerabilidades que explotar.
- *Denial of Service* (DoS): Ataque consistente en inundación de tráfico con fines de limitación de recursos de memoria y red para evitar la aceptación de peticiones legítimas.
- *User to Root* (U2R): Correspondiente a ataques en los que un usuario que ha iniciado una sesión es capaz de lograr privilegios de administrador.
- *Remote to Local* (R2L): Ataques en los que un usuario remoto sin acceso a un determinado equipo logra explotar una vulnerabilidad para iniciar una sesión.

Hay un total de 41 atributos que definen cada conexión, desde simbólicos, como el tipo de servicio (HTTP, SMTP, etc.) o el tipo de conexión de transporte (TCP, UDP, etc.), hasta continuos, como la longitud media de los paquetes de la conexión o el número de intentos de inicio de sesión. Algunos de los atributos son definiciones de alto nivel de otras trazas, como el número de conexiones al mismo equipo en una ventana de observación de dos segundos, estando incluidas por su relevancia demostrada en la detección de ataques [14].

Algunas características destacables en este conjunto de datos [15] son la existencia de patrones contradictorios (es decir, conexiones con clasificación diferente e igualdad de atributos) causados por situaciones en las que las características definidas son insuficientes para discernir el tipo de tráfico, así como la existencia de ataques diferentes en el conjunto de datos de prueba y el de entrenamiento, hecho que contribuye a dotar de realismo a la simulación dado que en un escenario real aparecerán nuevos patrones de ataque debido a las habilidades siempre cambiantes de los atacantes. En cualquier caso, todo nuevo patrón de ataque se considera encuadrable en uno de los diferentes grupos distinguidos (*normal*, *probe*, DoS, etc.), por lo que, obviando el ataque concreto, puede entenderse simplemente que el conjunto de datos de entrenamiento y el de pruebas contienen las mismos salidas (grupos de ataques) aunque, lógicamente, con diferentes ejemplares de conexiones diferenciados en mayor o menor medida.

Este tipo de conjuntos de datos de entrenamiento son escasos dada la dificultad de su elaboración, lo cual, unido a la naturaleza cambiante de los ataques posibles sobre un determinado sistema por su heterogeneidad y la continua mejora de las habilidades de los atacantes, constituye una de las principales dificultades para la implantación de sistemas de detección de intrusiones.

II-A1. Redes neuronales: Las redes neuronales son una técnica utilizada para llevar a cabo un aprendizaje automático de un conjunto de datos y obtener un sistema adaptativo a los cambios del entorno. Su utilización [16], [17] puede servir como clasificador automático de información si se disponen de conjuntos de datos supervisados para el entrenamiento del sistema. Dado que una de las principales dificultades de los sistemas de detección de intrusiones estriba en la naturaleza cambiante de los ataques posibles que se pueden realizar sobre un determinado equipo o sistema, las redes neuronales pueden automatizar la tarea de clasificación si se asume una supervisión de unos datos de entrada en una fase de entrenamiento del sistema.

Así, dichas facilidades de aprendizaje y adaptabilidad, pueden aplicarse sobre el aprendizaje de tráfico de red o de código [18]. En el primer caso, la utilización de una red neuronal permitiría, a partir de un conjunto de características extraídas del tráfico existente en una red, clasificar de forma automatizada ese tráfico en normal o en alguno de los diferentes tipos de ataque posibles para un determinado subsistema. En el segundo caso, de forma análoga se podría analizar el código de procesos en ejecución para, tras haber entrenado previamente una red neuronal con un conjunto de código de

programas con fines maliciosos, detectar automáticamente si en un determinado equipo se está ejecutando un programa no válido por haber sido víctima de un ataque.

En definitiva, las redes neuronales ofrecen la posibilidad de llevar a cabo un aprendizaje automático supervisado de un conjunto de datos con una consiguiente generalización de la solución. Para el caso que nos ocupa, poseen como inconvenientes el hecho de tener que asumir la responsabilidad de la supervisión de forma externa, de forma que en un escenario con un sistema monitorizado por un sistema de detección de intrusiones debería utilizarse una base de datos de entrenamiento en la que de forma manual se hubiese clasificado cada conjunto de datos en comportamientos admisibles o en alguno de los distintos tipos de ataque. Experimentalmente, además, se ha comprobado que presentan dificultades para generalizar ante nuevos ataques no presentes en el conjunto de datos de entrenamiento. Como ventaja, las redes neuronales permiten la mejora continua por aprendizaje, admitiendo la actuación de un operador humano que validase las detecciones y cancelase los falsos positivos para poder entrenar y mejorar iterativamente el sistema y mantener actualizadas sus capacidades de detección.

II-A2. Árboles de decisión: Los árboles de decisión [19] son un tipo de clasificador muy utilizado en sistemas de detección de intrusiones. Un árbol de decisión consta de nodos, arcos y hojas, de forma que cada nodo se etiqueta con un atributo y cada hoja con una clase, obteniéndose así un método de clasificación automática si se evalúan los diferentes nodos paso a paso con los atributos de un determinado ejemplar por clasificar.

Un árbol de decisión se construye durante el proceso de aprendizaje previo a la predicción de clases de nuevos ejemplares. Para el proceso de construcción del árbol, generalmente se emplean estrategias descendentes (de la raíz a las hojas), a través de la cual se infieren las diferentes reglas que definen el árbol de decisión, complementándose ocasionalmente este método con borrosidad y coeficientes de certidumbre en las reglas inferidas. Adicionalmente, se lleva a cabo un proceso de poda o eliminación de nodos más inferiores en caso de que no disminuyan la calidad de clasificación en el conjunto de entrenamiento con el fin de evitar sobreaprendizaje y que el clasificador generalice correctamente y se comporte de mejor forma ante nuevos ejemplares.

Los sistemas de detección de intrusiones basados en árboles de decisión [20], [16], de manera análoga a las redes neuronales, ofrecen unas prestaciones muy buenas aprendiendo patrones, pero fallan al generalizar dichos patrones a nuevos ataques. Pueden llevarse a cabo modificaciones que mejoren en gran medida las prestaciones de este clasificador [21]. Así, puede definirse una clase de ataque nueva en la que agrupar todos los patrones que no han sido identificados por las reglas en lugar de asumir dichos patrones como tráfico normal para mejorar de esta manera hasta en un 60 % la frecuencia de falsos negativos. Por ello y por el esquema de representación del conocimiento basado en reglas que poseen los árboles de decisión, fácilmente interpretable por un humano, son en muchos casos preferidos para su utilización en sistemas de detección de intrusiones frente a las redes neuronales.

II-A3. Mapas topológicos autoorganizativos: Los mapas topológicos autoorganizativos son una variante de las redes neuronales inspirada en fundamentos biológicos como es la existencia de una cierta interacción lateral entre neuronas, de forma que la salida de una determinada neurona influye a otras neuronas, introduciéndose el concepto de vecindad, por el cual las neuronas vecinas poseerán una cierta influencia entre ellas. El proceso de entrenamiento es no supervisado y competitivo, obteniéndose tras una fase de distribución global de neuronas seguida de otra de ajuste fino una caracterización del mapa con una aproximación a la función densidad de probabilidad de datos de entrada en forma de proyección sobre el mapa topológico de neuronas. Es decir, de forma automatizada y no supervisada, se obtiene un mapa autoorganizado con un mayor número de neuronas en las zonas con mayor densidad de probabilidad de los datos de entrada y con las neuronas localizadas en las entradas que han servido de entrenamiento. De esta manera, con un mapa topológico autoorganizativo pueden realizarse tareas de clasificación si, tras haber realizado el determinado entrenamiento, se asignan etiquetas en forma de clases a cada una de las neuronas del mapa en función de los conjuntos de datos de entrada que las activan. Para la tarea de clasificación, dada una determinada entrada existirá una neurona ganadora, seleccionada nuevamente por proximidad, cuya etiqueta determinará la clase correspondiente a la entrada, disponiéndose de mayor resolución neuronal en aquellas zonas del espacio de datos de entrada con mayor densidad de probabilidad.

Así, para el desarrollo de sistemas de detección de intrusiones [20], los mapas topológicos autoorganizativos permiten llevar a cabo tareas de clasificación de conjuntos de entrenamiento y prueba. Dado que la elaboración de conjuntos de datos de entrenamiento implica una tarea de supervisión manual muy costosa, los mapas topológicos son utilizados para llevar a cabo una primera clasificación en conjuntos de datos que serán utilizados como entrenamiento de un determinado clasificador. De esta manera, primeramente se entrenaría el mapa con los datos de entrada, después se etiquetarían las neuronas con la clase del ejemplar más próximo y, en la utilización del mapa, se aplicarían automáticamente a cada uno de los ejemplares la clase de su neurona más cercana. Finalmente, se buscaría un criterio de lejanía por el cual, para ciertos ejemplares del conjunto de entrenamiento, se exigiría una supervisión manual de su clase, al considerarse suficientemente alejados de su neurona ganadora, sirviendo, de esta forma, el mapa topológico autoorganizativo como ayuda en la clasificación y definición de conjuntos de datos de entrenamiento.

II-B. Seguridad en web

Las arquitecturas orientadas a servicios poseen nuevos protocolos, componentes y tecnologías con sus correspondientes posibles vulnerabilidades, de forma adicional a las vulnerabilidades típicas de toda tecnología basada en web [22]. Los servi-

cios web, como arquitectura distribuida, requieren protección a distintos niveles [23], buscándose la provisión de seguridad en comunicaciones y paso de credenciales, aseguración de la frescura, integridad y confidencialidad de mensajes, control de acceso y auditoría segura. Existen diversos esquemas de servicios web, como los basados en arquitecturas REST [24] o los estándares de servicios web del W3C [25], ofreciendo diferentes enfoques y tecnologías, pero paradigmas arquitectónicos semejantes.

Para la provisión de seguridad en los estándares de servicios web existe la especificación WS-Security [26]. Dicha especificación ofrece seguridad a nivel de mensajes y no de canal, como es realizado en otras arquitecturas.

Para ello, para cada envoltorio SOAP de una transacción se sigue un esquema modular mediante la inclusión de cabeceras de seguridad con resguardos o credenciales que aseguren autenticidad, estampados temporales que aseguren frescura, firmas que aseguren integridad y cifrado para asegurar la confidencialidad. En cuanto a los esquemas de autenticación, existen diversos mecanismos estandarizados para asegurar la interoperabilidad en composición de servicios, como Kerberos o SAML (*Security Assertion Markup Language*) [27].

De esta manera, se busca mitigar las principales amenazas a las que se enfrenta una arquitectura orientada a servicios [25], como alteración de mensajes, aseguración de confidencialidad, ataque de intermediario, suplantación de identidad, denegación de servicio o ataques de reenvío.

La estructuración de la red en forma de un conjunto de servicios para ejecutar las distintas acciones de negocio en forma de servicios disponibles para terceros supone un nuevo paradigma arquitectónico frente a anteriores enfoques. Existen diferentes tecnologías, algunas emergentes, que componen los estándares de servicios web y son empleadas por las arquitecturas orientadas a servicios [28]:

- XML (*Extensible Markup Language*): Formato básico de comunicación, extensible y validable mediante esquemas de etiquetas. Es utilizado en crudo por muchas implementaciones de arquitecturas REST al tiempo que sobre él se construyen estándares más complejos.
- SOAP (*Simple Object Access Protocol*): Es el protocolo de intercambio de mensajes en los servicios web, construido sobre XML.
- WSDL (*Web Service Description Language*): Lenguaje de descripción de servicios web, empleado para definir la funcionalidad de un servicio, su utilización y su interfaz y publicado en un registro UDDI (*Universal Description Discovery and Integration*).
- OWL (*Web Ontology Language*): En un nivel de abstracción superior, los estándares de comunicación de la web semántica se emplean en entornos de investigación para desarrollar servicios web semánticos y proporcionar mayor interoperabilidad y automatización a las arquitecturas orientadas a servicios a costa de una mayor complejidad.

Asociados con dichas tecnologías y con las propias arquitecturas orientadas a servicios, existe un número de ataques característico a los que se expone una organización que utilice servicios web [29]:

- Denegación de servicio: La carga de procesamiento que pueden requerir ciertos servicios web puede ser muy superior a la de otras aplicaciones web y, por tanto, soportar un menor número de peticiones por unidad de tiempo, debiéndose reajustar las herramientas de detección tradicionales para detectar ataques de denegación de servicio. Un caso similar es el ataque de reenvío, en el que los mensajes utilizados son mensajes previamente empleados y, por tanto, válidos en un principio.
- Desbordamiento de memoria: Las vulnerabilidades de intérpretes XML son heredadas, persistiendo la posibilidad de ejecutar código remoto ante la posibilidad de un desbordamiento de memoria por el envío de mensajes maliciosos.
- Ataque de diccionario: Los servicios web poseen sistemas de autenticación, siendo vulnerables a intentos de adivinación de contraseña.
- Inyección SQL: De forma análoga a las aplicaciones web, una arquitectura orientada a servicios utiliza datos intercambiados en consultas a base de datos, siendo vulnerables a ataques de inyección SQL para acceder a datos de forma no autorizada.
- *Cross-site scripting*: Por las tecnologías en las que se basan los servicios web, una arquitectura orientada a servicios es vulnerable a ejecución remota de código incluido en los datos intercambiados, en lo que se conoce como *cross-site scripting*.

En general, el paradigma de la seguridad en una arquitectura orientada a servicios es diferente a anteriores visiones en redes convencionales. La antigua diferenciación de servicios en función del número de puerto TCP/UDP no es suficiente para obtener información de relevancia de un servicio al tiempo que los actuales cortafuegos no diferencian servicios web puntuales al ignorar el identificador URI [23].

III. Sistema de detección de intrusiones

En este documento se propone un sistema de detección de intrusiones completo con un enfoque de aprendizaje automático para facilitar la actualización de la base de datos de reglas de detección.

III-A. Descripción del sistema

Se ha reutilizado el sistema de detección de intrusiones Snort [30], estándar de facto en su ámbito basado en software libre. Dicho sistema de detección de intrusiones utiliza una filosofía de Misuse Detection System o MDS, basada en firmas individualizadas para cada ataque e incluidas manualmente en una base de datos de reglas de disparo de alarmas, lo cual requiere un conocimiento experto por parte del agente humano encargado de dicha labor. Esto hace que Snort tenga capacidades de extensión bastante limitadas. Sin embargo, puede adaptarse dicha filosofía a un esquema adaptativo si se desarrolla un clasificador que aprenda a partir de distintos

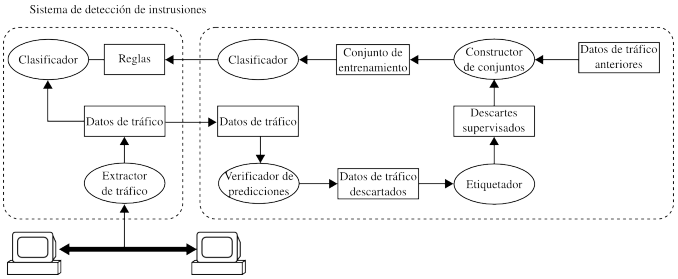

Figura 1. Arquitectura del sistema

ataques y genere las reglas apropiadas para la detección de dichos ataques.

De esta forma, se ha desarrollado un prototipo de sistema de detección de intrusiones. Mediante aprendizaje automático y supervisión asistida, el sistema busca agilizar el proceso de inclusión de reglas en Snort para favorecer la frescura en la detección de ataques por parte del sistema de detección de intrusiones.

En la tarea de detección de ataques y generación de reglas, se distinguen tres tareas: conocer para qué ataques el sistema no estaba preparado, clasificar dichos ataques y, finalmente, reentrenar el sistema generando nuevas reglas de detección. El sistema bajo descripción busca automatizar todas estas tareas en la medida de lo posible basándose en el mencionado sistema de detección de intrusiones basado en reglas Snort. La arquitectura del sistema se muestra en la figura 1, existiendo cuatro módulos principales [31]:

- Clasificador: Entrenado para clasificar muestras de tráfico y realizar las funcionalidades básicas de un sistema de detección de intrusiones.
- Verificador de predicciones: Valida las predicciones realizadas por el clasificador para detectar nuevos tipos de tráfico y realizar reentrenamiento con ellos.
- Etiquetador para clasificación de nuevos datos: Reduce el esfuerzo de clasificación manual de nuevos datos de tráfico agrupando muestras similares en conjuntos de un mismo tipo.
- Constructor de conjuntos de entrenamiento: Prepara un conjunto de datos para reentrenar el clasificador.

El sistema es, esencialmente, un generador de reglas, por lo que la clasificación de datos en tiempo real es realizada por el sistema de detección de intrusiones basado en reglas, mientras que los módulos adicionales llevan a cabo funciones en un segundo plano. Más concretamente, el proceso de generación de reglas comienza después de que un cierto conjunto de tráfico de datos haya sido recogido por el sistema de detección de intrusiones, procediéndose de la siguiente manera:

1. El verificador de predicciones estima la validez de las predicciones anteriores y construye un conjunto de datos de muestras descartadas, que requerirán supervisión adicional por la incapacidad del sistema de clasificarlas correctamente.
2. Los datos de tráfico son supervisados por un agente humano clasificando datos previamente etiquetados por el etiquetador.
3. El constructor de conjuntos de entrenamiento construye un nuevo conjunto con muestras de datos de tráfico anteriores y nuevas muestras recientemente supervisadas.
4. El clasificador es entrenado con el nuevo conjunto de entrenamiento. El resultado son nuevas reglas generadas, que son empleadas para refrescar la base de reglas del sistema de detección de intrusiones.

III-B. Aprendizaje de nuevo tráfico

Como ya se ha comentado, existen diversos enfoques para el aprendizaje de patrones de ataque en detección de intrusiones. Las redes neuronales han sido empleadas previamente [16], [17], pero presentan dificultades para generalizar su conocimiento y, por tanto, para detectar ataques que no están presentes en la base de datos de entrenamiento [21] . Otros enfoques han sido considerados, como modelos estadísticos [32], [33] o redes de Petri [34]. Ninguno de ellos puede ser usado de forma natural para construir reglas de detección y, por tanto, no son prácticos para nuestros propósitos.

Los árboles de decisión y los sistemas basados en reglas [19] también han sido utilizados para detección de intrusiones [20], [16], ofreciendo buenas prestaciones en términos de probabilidades de detección y generalización a nuevos ataques [21]. El sistema en descripción utiliza el algoritmo de aprendizaje de reglas C4.5 [35], que es esencialmente una extensión del algoritmo ID3 que pretende evitar el sobreaprendizaje.

Considerando un conjunto de datos de entrenamiento, dos tercios de él son utilizados como conjunto de datos de crecimiento del árbol, mientras que el tercio restante es utilizado como conjunto de poda. El conjunto de crecimiento es empleado para construir un árbol ID3, en el que una función de ganancia de entropía es utilizada para particionar el conjunto de datos respecto a los diferentes atributos.

El atributo con mayor ganancia de entropía es escogido para particionar el conjunto de datos en cada nodo, construyéndose un árbol de forma iterativa. Los atributos continuos son manejados de una forma equivalente, calculándose umbrales mediante interpolación de valores consecutivos del conjunto de datos para cada atributo continuo y escogiendo el umbral con mayor ganancia de entropía.

Después de que el árbol de decisión ha sido construido, la generación de reglas de inferencia es sencilla, consistiendo únicamente en recorrer todas las posibles rutas del árbol desde el nodo principal hasta las hojas. El lado izquierdo de las reglas será una combinación de las condiciones de cada nodo, mientras que el lado derecho consistirá en la clase mostrada en la hoja. Adicionalmente, se realiza una estimación de la precisión de cada regla calculando la precisión sobre el conjunto de poda. Las reglas resultantes son podadas eliminando las últimas condiciones del lado izquierdo de las reglas cuando la precisión estimada resultante no es menor. Finalmente, las reglas se ordenan por precisión estimada decreciente.

III-C. Verificación de predicciones

Utilizando el clasificador mencionado, se obtiene un conjunto de reglas con una estimación de precisión, que sirve como

factor de certidumbre en la predicción de clases de tráfico. En tiempo de detección, una determinada regla será activada, con su estimación de precisión asociada. En este punto, es posible forzar una estimación de precisión mínima para aceptar una predicción, siendo éste un heurístico que sirve para discernir datos de tráfico que fueron considerados en tiempo de entrenamiento frente a aquéllos que no. Por lo tanto, establecer un umbral de precisión permite poblar un conjunto de datos con datos que son supuestamente nuevos para el sistema y que, por tanto, requieren clasificación adicional. Como resultado, la predicción estimada de reglas permiten integrar capacidades de verificación de predicciones en el sistema.

III-D. Clasificación de nuevos datos

Una muestra es, por tanto, considerada nueva si el clasificador basado en reglas no es capaz de clasificarla apropiadamente como tráfico normal ni como ningún tipo de ataque, basando la decisión en un umbral de precisión estimada. De esta forma, estos datos necesitan supervisión manual por un agente externo para su clasificación. A pesar de ello, se puede proporcionar ayuda adicional en esta tarea si se agrupan de forma automática datos de tráfico similares. Para ello, el sistema bajo descripción utiliza mapas tolopológicos autoorganizativos, que han demostrado buenas prestaciones en anteriores estudios [11], [36]. Los mapas topológicos autoorganizativos [37] utilizan una métrica de similaridad euclídea para llevar a cabo agrupación automática de datos mediante la definición de un conjunto superpuesto de vectores de referencia en el espacio de características del conjunto de datos de entrenamiento. Se establecen relaciones de orden local sobre los vectores de referencia de forma que sus valores dependen de cada vector vecino. El algoritmo de autoorganización define una regresión no lineal de los vectores de referencia a través de los puntos de datos, que resulta en un reparto de conjuntos de referencia a lo largo del espacio en función de la función de densidad de probabilidad de las muestras. Esto permite clasificar todas muestras que son representadas por el mismo vector de referencia en un solo paso, reduciendo el esfuerzo de supervisión.

Al dimensionar el mapa topológico autoorganizativo, algunos problemas han de ser tratados, como escoger un número de nodos que haga capaz al mapa de adaptarse a todo el conjunto de datos o potenciar casos inusuales para que tengan representación y sean considerados por el mapa. Aplicar el algoritmo de autoorganización a todo el conjunto de datos daría lugar a una incapacidad por parte del mapa para adaptarse a valores demasiado separados en distancia euclídea e imposibilitando el tratamiento de casos inusuales que no son demasiado relevantes en la función densidad de probabilidad. Para evitar esto, la inspección visual de mapas de Sammon [38] aplicado a diferentes mapas ayuda a escoger una forma correcta o adaptar la función densidad de probabilidad, pero es una tarea manual que no se desea en el sistema, por lo que se utiliza un enfoque diferente. En este caso, el sistema realiza una división en varios subconjuntos del conjunto de descartes original para intentar obtener subconjuntos con características similares e incrementar la precisión del mapa topológico autoorganizativo. Se pueden utilizar diferentes heurísticos, como particionar mediante ciertos campos como el tipo de protocolo o el tipo de servicio [20], estando todos estos enfoques encaminados a reducir la entropía de información del subconjunto resultante. El conjunto de reglas del clasificador es una versión podada del árbol de decisión ID3 que, tal y como se describió previamente, se construye mediante reducción de entropía del conjunto de datos de entrenamiento, siendo, por tanto un heurístico posible para reducir la entropía en el conjunto de datos de descartes.

Para llevar a cabo esta subdivisión, las muestras se agrupan en el sistema por coincidencia jerárquica con las cláusulas de la regla. Más concretamente, cada muestra activa una determinada regla, cuyo lado izquierdo se define por una lista de cláusulas, y está ordenada por relevancia en la clasificación debido al algoritmo C4.5. Esto, por tanto, permite agrupar de forma jerárquica muestras similares eliminando las últimas cláusulas y agrupando todas las muestras que comparten las mismas cláusulas. Es preciso establecer un umbral de profundidad, de forma que un valor elevado produciría unn mayor número de subconjuntos, mientras que un valor más bajo produciría subconjuntos mayores con muestras más heterogéneas. La secuencia de cláusulas resultante se extiende con los campos de protocolo, tipo de servicio y flags para construir un identificador de subconjunto para cada muestra.

Finalmente, el algoritmo autoorganizativo se aplica en cada subconjunto. Se emplea una relación de aspecto de 3:2 en las dimensiones de los mapas para favorecer estabilidad en el aprendizaje, una topología hexagonal y un número total de nodos igual al 10 % de la cardinalidad de los subconjuntos con dimensión límite de 30x20.

III-E. Reentrenamiento

El algoritmo C4.5 no lleva a cabo entrenamiento por refuerzo. Para proporcionar aprendizaje con refuerzo, el enfoque utilizado en el sistema consiste en construir un nuevo conjunto de entrenamiento con diferentes proporciones de muestras. En este punto, tres tipos de muestras se encuentran en el sistema: muestras descartadas durante la verificación de predicciones, muestras del conjunto de datos de entrenamiento que se detectan correctamente y muestras del conjunto de datos de entrenamiento que no se detectan correctamente. La proporción de muestras de cada tipo y la cantidad total de ellas determinan el conjunto de datos de entrenamiento construido y, por tanto, las capacidades de clasificación del nuevo clasificador.

III-F. Evaluación

III-F1. Prestaciones del clasificador: El clasificador basado en reglas anteriormente descrito ha sido entrenado con un subconjunto del conjunto de datos de entrenamiento KDD'99 [13]. El clasificador presenta las dificultades conocidas para detectar ciertos ataques del conjunto de entrenamiento es consecuencia de su compromiso a generalizar a nuevos ataques [21]. Las prestaciones sobre el conjunto de datos de prueba

se muestran en la tabla I. En cualquier caso, las prestaciones globales son comparables con otros clasificadores, como el ganador de KDDCup'99 [39], un clasificador basado en votos que ofrece una precisión del 92'71 %.

Tabla I
PRESTACIONES SOBRE EL CONJUNTO DE PRUEBA.

Predicción / real	normal	probe	dos	u2r	r2l	Total
normal	99.49 %	17.76 %	2.76 %	54.29 %	90.79 %	73.29 %
probe	0.26 %	70.21 %	0.01 %	0.00 %	3.16 %	80.91 %
dos	0.22 %	12.03 %	97.22 %	0.00 %	0.03 %	99.72 %
u2r	0.02 %	0.00 %	0.00 %	35.71 %	2.62 %	5.38 %
r2l	0.01 %	0.00 %	0.00 %	10.00 %	3.40 %	95.52 %
Total	99.49 %	70.21 %	97.22 %	35.71 %	3.40 %	92.36 %

Tal y como se ha descrito previamente, un verificador de predicciones se emplea para descartar muestras potenciales cuyas predicciones podrían ser a priori consideradas inaceptables. Esto se lleva a cabo mediante una estimación de precisión de las reglas, calculada sober el conjunto de poda, como factor de confianza. El resultado del uso de diferentes umbrales de estimación de precisión A_{th} se muestra en la tabla II, de forma que, tal y como se esperaba, dicho umbral permite incrementar las prestaciones globales del clasificador. Esto permite mejorar cualquier otro clasificador realizado, con la contrapartida de marcar ciertas muestras conflictivas como descartadas. Observando los resultados, un umbral de precisión de 0'98 aparenta ser un valor apropiado al ofrecer un compromiso aceptable entre grado de descarte de paquetes y precisión.

Tabla II
EFECTO DEL UMBRAL DE PRECISIÓN.

A_{th}	Descartes	Precisión
0.0	0.00 %	92.36 %
0.9	1.13 %	93.11 %
0.95	1.59 %	93.19 %
0.96	1.59 %	93.19 %
0.97	1.59 %	93.19 %
0.98	1.83 %	93.21 %
0.99	5.73 %	94.07 %
0.995	5.73 %	94.07 %
0.999	5.73 %	94.07 %
0.9999	100.00 %	–

III-F2. Prestaciones del etiquetador: Las muestras descartadas son recogidas por el verificador de predicciones para supervisión adicional. En nuestro sistema, esta tarea la asiste el etiquetador de muestras. Las muestras descartadas obtenidas del clasificador se agrupan en un número de subconjuntos de acuerdo con los campos de las muestras y las cláusulas de las reglas activadas. Posteriormente, el algoritmo autoorganizativo se aplica sobre los subconjuntos, obteniéndose un número de nodos, siendo este número proporcional a la cardinalidad de los subconjuntos. Pueden utilizarse diferentes valores de profundidad en las cláusulas de las reglas para la subdivisión del conjunto de datos descartados original, obteniéndose diferentes resultados, tal y como se muestra en la tabla III, con valores más altos de precisión al emplear valores más altos de profundidad. Dado que el valor de profundidad más alto requiere la clasificación de únicamente el 15 % de las muestras y ofrece la precisión más alta, puede considerarse el valor de profundidad óptimo para el etiquetador.

III-F3. Prestaciones globales: Tras el etiquetado de paquetes descartados, el clasificador es reentrenado tras construirse un nuevo conjunto de entrenamiento. Son posbles

Tabla III
PRESTACIONES DEL ETIQUETADOR.

Profundidad	Subconjuntos	Nodos	Precisión
0	74	13.36 %	89.03 %
2	95	13.89 %	90.43 %
4	118	14.43 %	91.41 %
6	150	15.71 %	93.19 %
8	162	15.94 %	95.33 %

diferentes parámetros al construir el nuevo conjunto de entrenamiento. n_+ y n_- determinan la proporción de muestras de entrenamiento correcta e incorrectamente clasificadas, respectivamente, y ndis determina la proporción de muestras que fueron descartadas en tiempo de predicción. El resultado de variar dichos parámetros se muestra en la tabla IV. Las precisiones A_+ y A_- se calculan en los subconjuntos correctos e incorrectos del conjunto de entrenamiento, mientras que A_{dis} se calcula sobre el conjunto de datos descartados y etiquetados. La precisión global A se calcula sobre el conjunto de prueba completo.

Tabla IV
PRESTACIONES TRAS REENTRENAMIENTO.

n_+	n_-	n_{dis}	A_+	A_-	A_{dis}	A
0.5	0.4	0.1	99.71 %	100.0 %	92.55 %	92.32 %
0.5	0.1	0.4	99.64 %	100.0 %	94.61 %	93.32 %
0.3	0.2	0.5	99.47 %	100.0 %	95.08 %	93.41 %
0.4	0.1	0.5	99.56 %	100.0 %	94.94 %	93.46 %
0.5	0.3	0.2	99.54 %	100.0 %	93.45 %	93.53 %
0.3	0.1	0.6	99.57 %	100.0 %	94.94 %	93.63 %

Observando los resultados, el clasificador muestra un comportamiento óptimo con n_+ = 0'3, n_- = 0'1 y n_{dis} = 0'6. Globalmente, se demuestra una precisión muy buena para la restricción de emplear un sistema basado en reglas, añadiendo características de verificación de predicciones para recolección automática de datos problemáticos y clasificación asistida de datos de entrenamiento.

IV. CONCLUSIONES

Se ha propuesto un sistema de detección de intrusiones basado en Snort, que permite detectar intrusiones a partir de reglas generadas automáticamente, favoreciendo la frescura en la detección de ataques.

Tal y como se ha descrito, existen numerosos estudios que prueban la utilidad de técnicas de aprendizaje de datos para el aprendizaje automático de datos de tráfico y código para su clasificación automática en sistemas de detección de intrusiones. [40] es un trabajo similar, que busca la generación de firmas de ataques (como contraposición a la detección de vulnerabilidades), tratándose de un enfoque menos heurístico y generalizable y sin verificación de predicciones. [20] utiliza mecanismos diferentes para la verificación de predicciones, basado en votos y parámetros de confianza de clasificadores binarios, requiriendo diversas bases de datos de reglas e imposibilitando su implementación sobre Snort.

El sistema descrito propone una solución completa al problema de detección de intrusiones. Existen diversos trabajos enfocados a la optimización de las prestaciones de clasificadores. Sin embargo, se ha comprobado la existencia de acoplamiento entre los distintos módulos de un sistema de detección de intrusiones. La restricción de utilizar un sistema

basado en reglas condiciona el clasificador empleado, mientras que otros módulos presentan dependencias del clasificador, como la verificación de predicciones. En definitiva, considerando todo el ciclo presente en el mantenimiento de una base de datos de ataques, se ha propuesto un sistema de detección de intrusiones que reutiliza las características de estabilidad y altas prestaciones de sistemas basados en reglas ya desarrollados, como Snort.

AGRADECIMIENTOS

Parte de este trabajo de investigación ha sido financiado por el Gobierno Español para el proyecto IMPROVISA (TSI 2005-07384-C03-01).

REFERENCIAS

[1] B. Mukherjee, L. T. Heberlein, and K. Levitt, "Network Intrusion Detection," in *IEEE Network*, 1994.

[2] D. Denning, "An Intrusion-Detection Model," in *IEEE transactions on software engineering*, 1987.

[3] L. Deri, S. Suin, and G. Maselli, "Design and implementation of an anomaly detection system: An empirical approach," in *Proceedings of Terena TNC, 2003*, 2003.

[4] X. Zhang, C. Li, and W. Zheng, "Intrusion Prevention System Design," in *IEEE Computer and Information Technology CIT'04*, 2004.

[5] J. Botwicz, P. Buciak, and P. Sapiecha, "Building Dependable Intrusion Prevention Systems," in *Proceedings of the International Conference on Dependability of Computer Systems (DEPCOS-RELCOMEX'06)*, 2006.

[6] D. J. Ragsdale, "Adaptation Techniques for Intrusion Detection and Intrusion Response Systems," Information Technology and Operations Center, United States Military Academy, 2000.

[7] G. Vigna, W. Robertson, V. Kher, and R. A. Kemmerer, "A Stateful Intrusion Detection System for World-Wide Web Servers," in *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC 2003)*, 2003.

[8] C.-T. Lu, A. P. Boedihardjo, and P. Manalwar, "Exploiting Efficient Data Mining Techniques to Enhance Intrusion Detection Systems," Information Reuse and Integration, IEEE Conf, 2005.

[9] A. Siraj, R. B. Vaughn, and S. M. Bridges, "Intrusion Sensor Data Fusion in an Intelligent Intrusion Detection System Architecture," in *Proceedings of the 37th Hawaii International Conference on System Sciences*, 2004.

[10] B. Kim and I. Kim, "Kernel Based Intrusion Detection System," in *Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05), IEEE Computer Society*, 2005.

[11] N. Bashah and B. Shanmugam, "Artificial Intelligence Techniques Applied to Intrusion Detection," in *IEEE Indicon Conference, Chennai, India*, 2005.

[12] A. Vorobiev and J. Han, "Security Attack Ontology for Web Services," in *Proceedings of the Second International Conference on Semantics, Knowledge, and Grid (SKG'06), IEEE Computer Society*, 2006.

[13] University of California, "The Third International Knowledge Discovery and Data Mining Tools Competition Data," http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.

[14] W. Lee, S. Stolfo, and K. W. Mok, "A Data Mining Framework for Building Intrusion Detection Models," in *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1999.

[15] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan, "Cost-Based Modeling for Fraud and Intrusion Detection Results from the JAM Project," in *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX '00)*, 2000.

[16] S. Chavan, K. Shah, N. Dave, and S. Mukherjee, "Adaptive Neuro-Fuzzy Intrusion Detection Systems," in *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)*, 2004.

[17] R. A. Kemmerer and G. Vigna, "Hi-DRA: Intrusion Detection for Internet Security," in *Proceedings of the IEEE, October 2005*, 2005.

[18] S. Mandujano, A. Galván, and J. A. Nolazco, "An Ontology-based Multiagent Architecture for Outbound Intrusion Detection," in *IEEE Computer Systems and Applications*, 2005.

[19] E. B. Hunt, "Concept learning: an information processing problem," Wiley, 1962.

[20] Z. Yu, J. J. P. Tsai, and T. Weigert, "An Automatically Tuning Intrusion Detection System," IEEE Transactions on Systems, Man, and cybernetics, vol. 37, no. 2, April 2007, 2007.

[21] Y. Bouzida and F. Cuppens, "Neural networks vs. decision trees for intrusion detection," in *Proceedings of the 43rd annual Southeast regional conference*, 2005.

[22] OWASP, "The ten most critical web application security vulnerabilities," http://www.owasp.org/index.php/Top_10_2007, 2007.

[23] ——, "Web Services Security," http://www.owasp.org/index.php/Web_Services, 2007.

[24] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Univ. of California, Irvine, 2000. [Online]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[25] W3C, "Web Services Architecture," http://www.w3.org/TR/ws-arch/, 2004.

[26] OASIS, "OASIS Web Services Security (WSS) TC," http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss, 2006.

[27] ——, "OASIS Security Services (SAML) TC," http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security, 2007.

[28] M. Hondo, N.Nagaratnam, and A.Nadalin, "Securing Web Services," in *IBM Systems Journal, Vol. 41, no. 2*, 2002.

[29] Westbridge Technology, "Guide to XML Web Services Security," Technical report, 2003.

[30] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in *Proceedings of the 13th USENIX conference on System administration*, 1999.

[31] J. I. Fernández-Villamor and M. Garijo, "A Machine Learning Approach with Verification of Predictions and Assisted Supervision for a Rule-based Network Intrusion Detection System," in *Proceedings of the fourth International Conference on Web Information Systems and Technologies*, In-press.

[32] N. Ye, S. Emran, X. Li, and Q. Chen, "Statistical Process Control for Computer Intrusion Detection," in *Proceedings DISCEX II*, 2001.

[33] N. Ye, S. Vilbert, and Q. Chen, "Computer Intrusion Detection through EWMA for Auto Correlated and Uncorrelated Data," in *IEEE Transactions on Reliability*, 2003.

[34] S. Kumar and E. Spafford, "A Pattern Matching Model for Misuse Intrusion Detection," in *Proceedings of the 17th National Security Conference*, 1994.

[35] R. Quinlan, "C4.5: Programs for Machine Learning," Morgan Kaufmann Publishers, Inc., 1993.

[36] A. J. Hoglund, K. Hatonen, and A. S. Sorvari, "A Computer Host Based User Anomaly Detection System Using Self Organizing Maps," in *Proceedings of the International Joint Conference on Neural Networks, IEEE IJCNN 2000, Vol. 5, pp. 411-416*, 2000.

[37] T. Kohonen, "Self-Organizing Maps," Springer-Verlag New York, Inc., 1997.

[38] J. W. Sammon, "A nonlinear mapping for data structure analysis," IEEE Transactions on Computers, C-18(5):401-409, May 1969, 1969.

[39] B. Pfahringer, "Winning the KDD99 classification cup: Bagged boosting," in *ACM SIGKDD Explor., vol. 1, no. 2, pp 65-66*, 1999.

[40] L. C. Wuu and S. F. Chen, "Building Intrusion Pattern Miner for Snort Network Intrusion Detection System," in *IEEE Computer Society*, 2003.