RNPS No. 2143
ISSN 2072-6260
Versión Digital

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada

REPORTE TÉCNICO
Minería de Datos

SERIE GRIS

Modelos basados en reglas de clasificación para la detección de actividades maliciosas

Vitali Herrera-Semenets, Osvaldo Andrés Pérez-García, Andrés Gago-Alonso y Raudel Hernández-León

RT_035 **julio 2016**

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

RNPS No. 2143
ISSN 2072-6260
Versión Digital

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada

REPORTE TÉCNICO
Minería de Datos

SERIE GRIS

Modelos basados en reglas de clasificación para la detección de actividades maliciosas

Vitali Herrera-Semenets, Osvaldo Andrés Pérez-García, Andrés Gago-Alonso y Raudel Hernández-León

RT_035 **julio 2016**

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

Siglas y acrónimos

ADES: Descriptores aceptados (en inglés *Accepted DEScriptors*).
AM: Actividades Maliciosas.
AQ21: Algoritmo de inducción natural.
C5.0: Algoritmo que permite generar reglas basándose en los árboles de decisión.
CART: Algoritmo que permite la construcción de un árbol de clasificación y regresión.
FURIA: Algoritmo de inducción de reglas difusas sin ordenar (en inglés *Fuzzy Unordered Rule Induction Algorithm*).
GAR: Generación Automática de Reglas.
HTTP: Protocolo de transferencia de hipertexto (en inglés *Hypertext Transfer Protocol*)
NDES: Descriptores negativos (en inglés *Negative DEScriptors*).
PART: Algoritmo que permite generar reglas basándose en los árboles de decisión.
PDES: Descriptores potenciales (en inglés *Potential DEScriptors*).
QoS: Calidad de Servicio (en inglés *Quality of Service*).
RIPPER: Algoritmo que permite generar reglas basándose en los árboles de decisión (en inglés *Repeated Incremental Pruning to Produce Error Reduction*).
TIC: Tecnologías de la Información y las Comunicaciones.
X2R: Algoritmo de generación de reglas basado en discretización y selección de características.

Notaciones

c_l	l-ésima clase
C	Conjunto de clases
D	Colección de entrenamiento
E	Conjunto de aristas
F_i	i-ésimo conjunto de atributos
f_i	i-ésimo atributo
G	Grafo
H	Límite de Hoeffding
I	Intervalo
L_r	Estructura de datos de la regla r
m	Condición
n	Instancia
i, k, l	Subíndices (números enteros no negativos)
r	Reglas
r_i	i-ésima regla
$\overleftarrow{r}$	Premisa
$\overrightarrow{r}$	Condición de igualdad
R	Conjunto de reglas
s	Subgrafo
S	Colección de subgrafos
T	Conjunto de árboles
U	Universo
v	Vértice
V	Conjunto de vértices
W	Ventana deslizante
x, y	Límites de un intervalo
$\oplus$	Operador relacional
ν	Valor del dominio de f_i
λ	Función de pertenencia
ϱ	Medida de crecimiento
ι	Medida de intersección entre dos reglas

Tabla de contenido

Lista de figuras

Lista de tablas

Modelos basados en reglas de clasificación para la detección de actividades maliciosas

Vitali Herrera-Semenets, Osvaldo Andrés Pérez-García, Andrés Gago-Alonso y Raudel Hernández-León

Equipo de Investigaciones de Minería de Datos, CENATAV - DATYS,
{vherrera, osvaldo.perez, agago, rhernandez}@cenatav.co.cu

RT_035, Serie Gris, CENATAV - DATYS
Aceptado: 6 de julio de 2016

Resumen. Las entidades que brindan servicios basados en las Tecnologías de la Información y las Comunicaciones (proveedores de acceso a Internet, telefonía fija y móvil, entre otros) pueden ser afectadas por actividades maliciosas, las cuales causan pérdidas millonarias y dañan la disponibilidad de los servicios afectando el prestigio de dichas entidades. Con el objetivo de evitar tales daños, es necesario analizar flujos de eventos generados por la prestación de servicios. Los flujos de eventos tienen características especiales, tales como altas velocidades y grandes cantidades de datos, así como diversidad de fuentes y formatos. Teniendo en cuenta esto, para el análisis de dichos flujos, se hace necesario el uso de modelos efectivos que puedan ser utilizados en tiempo real para detectar actividades maliciosas en el menor tiempo posible. Los modelos basados en reglas son reportados como uno de los más utilizados para la detección de actividades maliciosas. En este trabajo se discuten varios modelos basados en reglas de clasificación para la detección de actividades maliciosas. Además, se describen esquemas generales para una mejor comprensión de cada modelo. Por último, se presentan los problemas identificados en dichos modelos.

Palabras clave: detección de actividades maliciosas, generación de reglas, minería de datos.

Abstract. Entities providing services based on Information and Communications Technologies (Internet access providers, landline and mobile, among others) are targets of malicious activities that cause millions in losses and affect their prestige. In order to prevent such damage, it is necessary to analyze event streams generated by service provision. Event streams have special features, such as high speeds and large amounts of data, as well as diversity of sources and formats. Therefore, the use of effective models that can be used in real time are required. Rule-based models are reported as one of the most used for malicious activities detection. In this paper are discussed several classification rule-based models for malicious activities detection. For a better understanding of each model, their general schemes are outlined. Finally, identified problems in the models are presented.

Keywords: malicious activities detection, rule generation, data mining.

1. Introducción

En la actualidad existen entidades tales como compañías telefónicas, bancos, entre otras, que brindan servicios basados en las Tecnologías de la Información y las Comunicaciones (TIC). La ejecución de actividades maliciosas (AM) mediante dichos servicios causa pérdidas millonarias [1],

daña la disponibilidad de los servicios y afecta la imagen de las entidades afectadas. Las actividades maliciosas incluyen acciones tales como: (1) fraudes en servicios de telecomunicaciones, (2) intrusiones en redes de telecomunicaciones y (3) fraudes en transacciones bancarias.

Las entidades que prestan servicios basados en las TIC, requieren técnicas automatizadas para detectar en el menor tiempo posible eventos asociados a la ocurrencia de actividades maliciosas [2]. Estas técnicas deben ser diseñadas para funcionar en escenarios complejos (donde se manifiestan las AM) caracterizados por:

- Flujos de datos con elevada cantidad de instancias, pudiendo alcanzar el orden de los millones en minutos (intrusiones en redes de telecomunicaciones).
- Instancias con elevado número de atributos.
- Alto número de clases asociadas a diferentes tipos de AM.
- Los datos que describen actividades maliciosas pueden ser modificados con el objetivo de hacer fallar al método de detección (cambio de concepto).

Para caracterizar el rendimiento de los métodos sobre este tipo de escenarios se utilizan parámetros como la precisión y la velocidad de detección [3]. Una amplia variedad de técnicas han sido propuestas para la detección de actividades maliciosas [4,5]. Dichas técnicas se basan sobre modelos específicos, de los cuales tres son muy utilizados en la práctica: los basados en reglas de clasificación, los basados en anomalías y los híbridos.

El modelo basado en reglas de clasificación requiere de un conocimiento previo del dominio para el proceso de generación de reglas, lo cual lo hace un modelo supervisado (ver Figura 1). El proceso de generación de reglas puede ser llevado a cabo de forma manual por un analista o puede ser ejecutado de forma automática mediante métodos de minería de datos. Cuando el proceso se realiza de forma manual, el analista crea las reglas basándose en actividades maliciosas conocidas. Por otra parte, la generación automática de reglas (GAR) utiliza un conjunto de entrenamiento etiquetado, donde cada instancia es etiquetada con su respectiva clase (la clase puede ser *normal* o algún tipo de actividad maliciosa). Dicho conjunto de entrenamiento es procesado por un método donde la salida es un conjunto de reglas. Las reglas generadas son evaluadas en tiempo real y en caso de cumplirse alguna regla se emite una alerta. El modelo basado en reglas tiene ciertas ventajas, como son: (1) alta efectividad sobre AM conocidas, (2) pueden detectar AM en tiempo real y (3) proveen una mejor comprensión a los analistas sobre cómo se describen las actividades maliciosas. Por otra parte, los principales inconvenientes identificados en este modelo son: (1) no puede detectar cambios sutiles en los datos y (2) el conjunto de reglas tiene que ser actualizado con frecuencia para poder detectar nuevas AM.

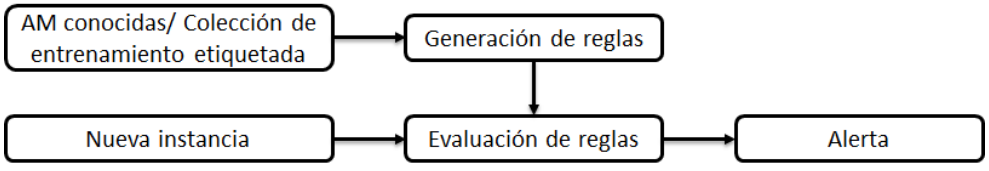

Fig. 1. Modelo basado en reglas de clasificación para la detección de actividades maliciosas.

El modelo basado en anomalías detecta acciones anómalas a partir de un comportamiento normal establecido (ver Figura 2). El comportamiento normal se define mediante un conjunto de entrenamiento no etiquetado que refleja la información histórica. Luego de tener definido el comportamiento normal, este puede ser comparado con respecto al comportamiento actual, lo

cual permite determinar si han tenido lugar cambios significativos que indiquen la ocurrencia de una anomalía. Entre las ventajas de este modelo, se destacan: (1) la posibilidad de detectar cambios sutiles en el comportamiento de los suscriptores y (2) no es necesario un conocimiento previo del dominio, lo cual permite identificar nuevas y desconocidas actividades maliciosas. Por otra parte, los inconvenientes incluyen: (1) las anomalías son asociadas a AM (aumentan los falsos positivos) y (2) las actividades maliciosas no pueden detectarse en tiempo real.

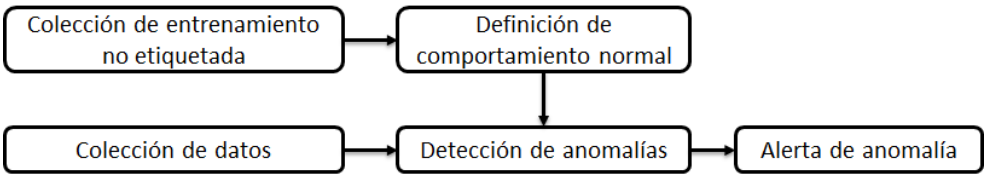

Fig. 2. Modelo basado en anomalías para la detección de actividades maliciosas.

También existen métodos basados en el modelo híbrido [6,7], que consiste en una combinación de los dos modelos presentados anteriormente. El modelo híbrido utiliza la detección de anomalías para etiquetar el conjunto de entrenamiento de manera no supervisada [8]. Luego, utilizando el conjunto ya etiquetado, se generan reglas que van a definir anomalías. Las reglas generadas, al igual que en el modelo basado en reglas, son evaluadas en tiempo real. Las ventajas que presenta este modelo son: (1) las anomalías se pueden detectar en tiempo real, (2) no se requiere un conocimiento previo del dominio y (3) proporciona una mejor comprensión a los analistas sobre cómo se definen las anomalías. Las desventajas son consistentes con algunos de los problemas identificados en los modelos anteriores, entre los que se incluyen: (1) las anomalías son asociadas a AM (aumentan los falsos positivos) y (2) el conjunto de reglas necesita ser actualizado con frecuencia para detectar nuevas anomalías.

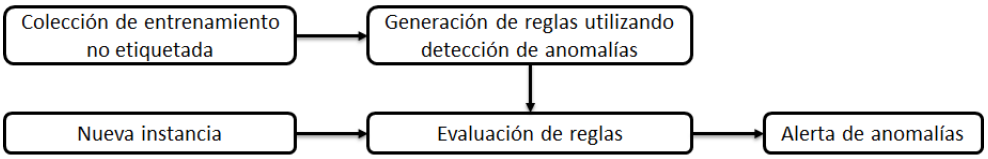

Fig. 3. Modelo híbrido para la detección de actividades maliciosas.

A pesar de los avances tecnológicos que proporcionan mayor velocidad para el análisis de datos y más rigor en el acceso a los servicios, la detección y prevención de actividades maliciosas sigue siendo un reto fundamental para la mayoría de las entidades que brindan servicios basados en las TIC. Vale resaltar que más del 50 % de las actividades maliciosas se detectan después de haber producido daños [9]. Además, en muchos casos, los modelos utilizados para asistir a los analistas en la detección de AM, generan un alto número de falsas alarmas afectando el correcto flujo de información y en consecuencia la calidad de servicio (QoS, por sus siglas en inglés). Teniendo en cuenta esto, las entidades requieren de un modelo para detectar AM en tiempo real, que haga posible evitar o minimizar los daños que puedan ocasionar las AM. Además, dicho modelo debe funcionar sin afectar la calidad de servicio que proveen las entidades.

Durante el proceso de detección de AM en tiempo real, cuando se genera una alerta, la actividad implicada es bloqueada con el fin de evitar daños. En este sentido, los modelos tales como los basados en anomalías e híbridos tienen desventajas en común. Dichos modelos asocian

las anomalías a actividades maliciosas, cuando una anomalía pudiera representar un nuevo comportamiento normal. Además, en estos modelos las AM semejantes al comportamiento normal pueden pasar desapercibidas, y por último, no explotan el conocimiento existente asociado a ataques conocidos [10]. Esto se traduce en un aumento del número de falsos positivos y hace que las actividades normales sean bloqueadas afectando la QoS de la entidad. Por otro lado, el modelo basado en reglas de clasificación utiliza el conocimiento existente del dominio para definir reglas representativas de AM, haciendo poco probable que se generen falsos positivos. Por esta razón, los modelos basados en reglas de clasificación suelen ser utilizados ampliamente en los escenarios descritos anteriormente.

En este trabajo se presenta una revisión de los modelos basados en reglas de clasificación para la detección de actividades maliciosas. El propósito es analizar dichos modelos y determinar rasgos que puedan afectar su efectividad durante el proceso de detección de actividades maliciosas. Este trabajo consta de cinco secciones. En la actual sección, se presentó una introducción al tema que se aborda en este artículo. Seguidamente, se expondrán algunos conceptos básicos. En la tercera sección, se analizan varios modelos basados en reglas de clasificación para la detección de actividades maliciosas. En la cuarta sección, se discuten los problemas identificados en los modelos analizados. Por último, en la quinta sección se presentan las conclusiones generales de este trabajo.

2. Conceptos básicos

En esta sección se presentan algunos conceptos básicos para un mejor entendimiento de los modelos analizados más adelante.

2.1. Calidad de servicio (QoS)

La calidad de servicio (QoS, por sus siglas en inglés) en el campo de las TIC, se puede definir como un conjunto de requisitos específicos de una red de telecomunicaciones que son necesarios para lograr la funcionalidad requerida de un servicio [11]. Los parámetros y las medidas de calidad de servicio se utilizan como un indicador para determinar el comportamiento de un servicio. Cada entidad que proporciona servicios basados en las TIC es responsable de garantizar cierto nivel de rendimiento para el flujo de datos, lo cual define la calidad del servicio ofrecido por dicha entidad.

2.2. Regla de clasificación

Para comprender en que consiste una regla es necesario definir primero algunos elementos. Sean $F_1, F_2, ..., F_k$ conjuntos de atributos. Una instancia $n = (f_1, f_2, ...f_k)$ puede ser definida como un vector perteneciente al universo $U = F_1 \times F_2 \times ... \times F_k$, donde $f_i \in F_i$ para cada $1 \leq i \leq k$.

Teniendo en cuenta esto, una regla de clasificación r en el universo $U \times C$ es representada por $r = \langle \overleftarrow{r}, \overrightarrow{r} \rangle$, donde $C = \{c_1, c_2, ..., c_l\}$ es un conjunto de clases con $l \geq 2$ siendo l un entero, la variable $\overleftarrow{r}$ representa la premisa de r y $\overrightarrow{r}$ es una condición de igualdad definida sobre la componente correspondiente a las clases (C). La premisa $\overleftarrow{r}$ es una conjunción de condiciones. Una condición m puede ser representada como $f_i \oplus \nu$, done f_i es un atributo condicional, la variable ν es un valor del dominio de f_i y $\oplus$ representa un operador relacional del conjunto de

relaciones $\{<, \leq, =, \neq, >, \geq, \in\}$. Por simplicidad, una regla suele ser representada como $\langle \overleftarrow{r}, c_i \rangle$, donde $c_i \in C$. El significado intuitivo de una regla consiste en que el cumplimiento de la premisa $\overleftarrow{r}$ en una instancia implica que dicha instancia pertenece a la clase c_i.

2.3. Reglas difusas

Las reglas difusas analizadas en este trabajo constituyen un tipo especial de reglas de classificación $\langle \overleftarrow{r}, c_i \rangle$, donde las condiciones que conforman la premisa difieren un tanto de las utilizadas en las reglas de clasificación. Las reglas no difusas contienen condiciones con límites exactos, mientras las reglas difusas contienen condiciones con límites flexibles. Por ejemplo, una condición que restringe un atributo numérico f_i (con dominio $\mathbb{D}_i = \mathbb{R}$) puede ser expresada en la forma $(f_i \in I)$, donde $I \in \mathbb{R}$ es un intervalo: $I = (-\infty, x]$ si la regla contiene una condición $(f_i = x)$, o $I = [y, \infty)$ si la regla contiene la condición $(f_i = y)$. De esta forma, un intervalo puede definirse utilizando una función de pertenencia λ.

2.4. Metodología CRISP-DM

Existen metodologías como KDD [12], SEMMA [13] y CRISP-DM [14] para la aplicación de técnicas de minería de datos, lo cual permite establecer relaciones entre diferentes procesos para formar un modelo eficiente y eficaz. Según la encuesta publicada en [15], existe una preferencia a utilizar la metodología CRISP-DM en proyectos de minería de datos. CRISP-DM fue propuesta por Chapman *et al.* [14]; su uso contribuye en la planificación y ejecución de modelos de minería de datos. Dicha metodología es considerada un proceso estándar para la aplicación de técnicas de minería de datos en el análisis de grandes volúmenes de datos (Big Data) [16]. Como se muestra en la Figura 4 (tomada de [17]), existen seis fases [17] que definen el ciclo de vida de un proyecto de minería de datos: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación e implementación. Las flechas indican las dependencias más importantes y frecuentes entre fases. En un proyecto en particular, dichas flechas pueden indicar en qué orden debe ejecutarse cada fase.

3. Modelos basados en reglas de clasificación para la detección de actividades maliciosas

Los modelos basados en reglas de clasificación representan una solución efectiva para el análisis de datos en tiempo real. En los escenarios descritos anteriormente, la principal desventaja de la generación manual de reglas consiste en que es el analista quien debe definir reglas específicas para cada clase existente, lo cual es una tarea extremadamente compleja y requiere mucho tiempo [18]. Ante esta desventaja, la generación automática de reglas ha comenzado a jugar un papel fundamental en la modelos basados en reglas de clasificación para la detección de actividades maliciosas [19].

A partir del estudio de los modelos que se abordan en esta sección fue posible definir un modelo general (ver Figura 5). El modelo general requiere una colección de entrenamiento correctamente etiquetada, en la cual las instancias que la componen deben pertenecer a la clase que las define como maliciosa o legítima. El próximo paso consiste en el preprocesamiento de datos para reducir

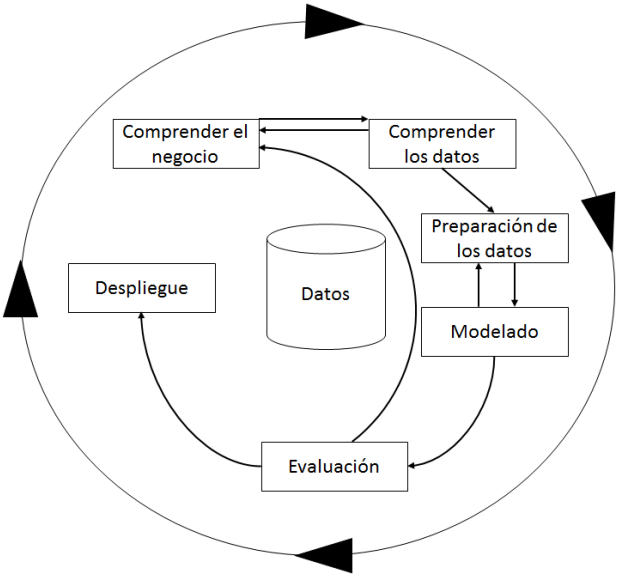

Fig. 4. Fases del modelo de procesos CRISP-DM.

la colección de entrenamiento y mejorar su calidad eliminando datos que generan ruido [20]. Luego, la colección de datos obtenida es procesada por un algoritmo de generación de reglas [19]; estos crean un conjunto de reglas para ser evaluadas sobre nuevas instancias. Cuando se cumple una regla durante el proceso de evaluación, se lanza una alerta a la interfaz de usuario, donde el analista es notificado. Mediante la interfaz de usuario, el analista puede comprobar cómo se están comportando las reglas, lo cual puede ser un indicador para determinar si se debe preparar una nueva colección de entrenamiento para actualizar las reglas existentes.

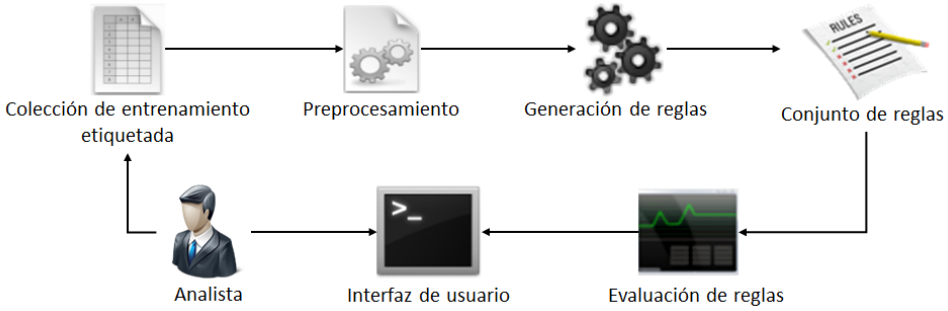

Fig. 5. Modelo general basado en reglas de clasificación para la detección de AM.

Teniendo en cuenta las características de los escenario donde tienen lugar las AM, la aplicación de un modelo cíclico pudiera resultar más efectiva en la práctica, ya que la minería de datos no terminaría con el proceso de evaluación. De esta forma, los resultados obtenidos durante cada etapa del modelo y de la solución desplegada pueden contribuir a mejorar el resultado final, beneficiándose cada etapa con la experiencia de las anteriores. Este hecho hace que el modelo general presente cierta semejanza con el modelo de proceso CRISP-DM.

Los métodos basados en GAR detectan patrones de actividades maliciosas y los representan como reglas. Esta forma de representar los patrones proporciona un mejor entendimiento a los

analistas de cómo se describen las AM, ya que las salidas están definidas como expresiones en un lenguaje comprensible, a diferencia de otros métodos tales como los basados en detección de anomalías que son considerados como cajas negras [8].

3.1. Modelo basado en grafos

La expresividad e idoneidad de los grafos, han contribuido a que estos sean ampliamente utilizados para modelar datos en diversos contextos de aplicación, dentro de los que se incluye la detección de actividades maliciosas. Una técnica para asegurar la protección en votaciones electrónicas fue presentada por Djanali *et al.* [21]. En esta técnica se utiliza un modelo basado en grafos (ver Figura 6) que permite generar reglas a partir de registros del protocolo HTTP.

Con el fin de obtener registros de solicitudes HTTP, se utilizó un honeypot. Un honeypot es una herramienta de seguridad informática utilizada para recoger información sobre los atacantes y sus técnicas [22].

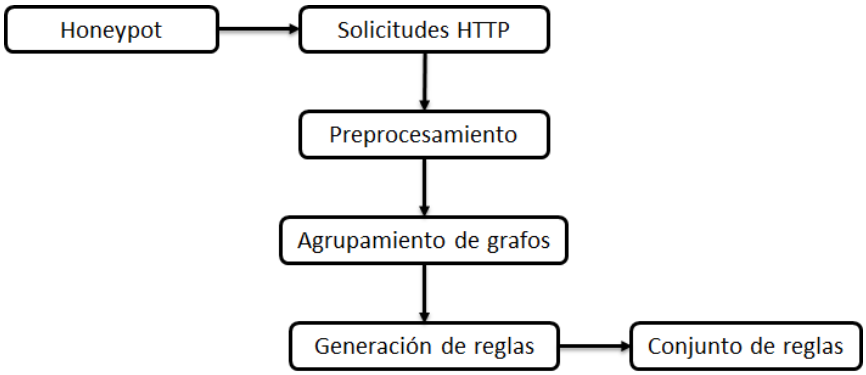

Fig. 6. Modelo basado en grafos.

Las solicitudes HTTP recogidas por el honeypot se preprocesan para garantizar que no contengan caracteres innecesarios. El siguiente paso consiste en realizar un agrupamiento de grafos. Para ello, las solicitudes HTTP se almacenan en un grafo G $= \{V, E\}$, donde V es un conjunto de vértices y E es un conjunto de aristas que conecta a los vértices. Cada vértice $v \in V$ denota una petición entrante. Una vez creado un nuevo vértice, se lleva a cabo una comparación con los vértices existentes para determinar su similaridad con respecto a los existentes. La distancia entre dos vértices viene dada por la distancia euclidiana y su valor se utiliza como la etiqueta de la arista que los conecta. Dos vértices estarán conectados si su distancia es menor que la de un umbral predeterminado o si está por debajo de la anterior menor distancia del vértice. El grafo resultante G puede ser un grafo no conexo, con varios subgrafos completamente separados $S \in G$.

El siguiente paso consiste en la generación de reglas. En este proceso, el grafo G es podado utilizando el algoritmo de árbol de expansión mínima, donde se procesan varios subgrafos S. Después de la poda, se genera una regla para cada subgrafo $s \in S$. Para ello, primero se busca cada vértice raíz. Luego, se recorre cada subgrafo s asociado a los vértices raíz. Cuando se visita un vértice, los datos que contiene de la solicitud HTTP se toman como una cadena. A partir de todas las cadenas tomadas, se busca la subcadena común más larga. Finalmente, utilizando la subcadena común más larga encontrada se procede a generar una regla.

Estos pasos se repiten periódicamente. Cada vez que llegue una nueva solicitud HTTP, esta se añadirá directamente al grafo existente.

3.2. Modelo de búsqueda voraz

En esta sección, se describen los métodos basados en el modelo de búsqueda voraz (AQ21 [23], X2R [24], RIPPER [25], PART [26] y Ant-Miner [27]). Como se muestra en la Figura 7, el primer paso consiste en preprocesar la colección de entrenamiento. Este paso no es obligatorio dentro del modelo voraz, ya que la colección de entrenamiento pudiera estar lista para ser procesada en los siguientes pasos. Sin embargo, algunos de los métodos analizados utilizan esta función para discretizar la colección de entrenamiento (X2R) o asignar un valor inicial a determinadas variables de estado (RIPPER).

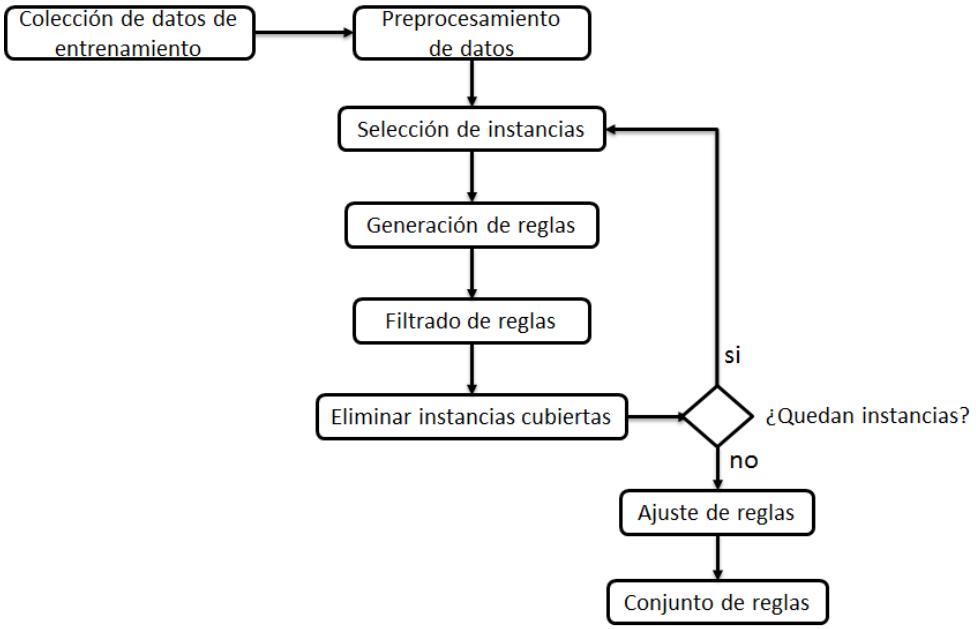

Fig. 7. Modelo de búsqueda voraz.

Sobre la colección de entrenamiento se ejecuta un proceso iterativo. Generalmente, el criterio de parada se basa en comprobar si todavía quedan instancias en la colección para ser procesadas. No obstante, existen métodos voraces que incorporan otras estrategias de parada además de la mencionada anteriormente; por ejemplo: AQ21 también comprueba si la colección de entrenamiento contiene instancias positivas (en nuestro caso, las instancias que representan actividades maliciosas), y RIPPER además verifica que la última regla encontrada no sea muy complicada (el grado de complejidad de una regla se da en términos de la longitud total de la descripción [28]).

En cada iteración, se selecciona un subconjunto de la colección de entrenamiento. Utilizando dicho subconjunto se genera un primer conjunto de reglas candidatas. Seguidamente, el conjunto de reglas obtenidas es filtrado, descartando las que en menor medida representan AM. Al final de cada iteración, aquellas instancias que son cubiertas por las reglas filtradas se eliminan de la colección de entrenamiento. En la Tabla 1, se presenta un resumen de las estrategias de iteración

utilizadas por diferentes métodos, lo cual muestra diferentes formas de implementar los siguientes procesos: selección de instancias, generación de reglas y filtrado de reglas.

Tabla 1. Resumen de estrategias utilizadas por métodos basados en el modelo de búsqueda voraz.

	AQ21	Ant-Miner	PART	X2R	RIPPER
Selección de instancias	Selecciona una instancia positiva n llamada semilla (en inglés *seed*).	No se ejecuta el proceso de selección, por tanto, la colección de entrenamiento original se utiliza en los restantes procesos.		Selecciona un subconjunto con las instancias más frecuentes en la colección de entrenamiento.	Selecciona aleatoriamente 2/3 de las instancias de la colección de entrenamiento.
Generación de reglas	Obtiene un conjunto de reglas (estrella aproximada) para la semilla que no cubren ninguno de los ejemplos negativos.	Calcula un conjunto de reglas empleando un método basado en colonia de hormigas [29] (ACS por sus siglas en inglés).	Calcula el árbol de desición simplificado asociado a la colección de entrenamiento, usando el algoritmo C4.5 [30]. Luego, se busca la hoja que más instancias cubre y se determina el camino que conduce desde la raíz hasta dicha hoja. El camino obtenido es representado en forma de regla.	Crea una regla vacía y le adiciona condiciones que son aprendidas utilizando la colección de entrenamiento hasta que la regla no cubra más instancias de otras clases. Cada condición es adicionada de forma tal que se maximice alguna medida de calidad, definida según el contexto de aplicación.	
Filtrado de reglas	Selecciona las reglas de acuerdo a una medida de calidad, definida según el contexto de aplicación.	Selecciona solamente la mejor regla descartando las restantes, de acuerdo a una medida de calidad, definida según el contexto de aplicación.	No se realiza ninguna acción.		Reemplaza la regla por una más general, eliminando condiciones de la premisa.

Después de haber completado todas las iteraciones, los métodos voraces pueden realizar un último proceso de filtrado sobre el conjunto final de reglas. Este último paso no es obligatorio, sin embargo, algunos métodos (como AQ21) utilizan medidas como el funcional de evaluación lexicográfica [23] para eliminar reglas del conjunto final; mientras que otros (X2R y RIPPER) eliminan reglas redundantes de cada clase y reemplazan las reglas específicas por otras más generales.

3.3. Modelo de reglas difusas

La lógica difusa a menudo se puede utilizar para generar un conjunto de reglas difusas aproximadas para la detección de actividades maliciosas. Por ejemplo, Barthakur *et al.* [31], proponen un modelo de reglas difusas para la detección de botnets en una red de pares (P2P). Una botnet es un conjunto de computadoras (o robots) controladas de forma remota por un individuo u organización, con el objetivo de atacar sistemas informáticos [32].

Inicialmente, la colección de entrenamiento es preprocesada, donde se realiza un análisis detallado del comportamiento del flujo de tráfico de los botnets. Posteriormente, los atributos más significativos para la clasificación se seleccionan de los encabezados de los paquetes.

El preprocesamiento garantiza la reducción de la dimensionalidad de los datos para ser procesados en la etapa de generación de reglas. En esta etapa se genera un conjunto de reglas difusas utilizando el algoritmo FURIA [33]. En este algoritmo se realiza una adaptación del algoritmo RIPPER [25], el cual es utilizado para calcular un primer conjunto de reglas. Estas reglas son transformadas en reglas difusas mediante la modificación de las condiciones asociadas a atributos numéricos. Las reglas difusas generadas por FURIA cumplen con las definiciones planteadas en la Sección 2. En este caso, la función de pertenencia λ es definida como una función trapezoidal [33]. Además, la premisa en estas reglas se conforma por la conjunción lógica de una o varias condiciones. Por tanto, el cubrimiento de una instancia viene dado por el producto de todos los valores de las condiciones que componen la premisa.

3.4. Modelo de árboles de decisión

Existen varios métodos basados en el modelo de árboles de decisión, algunos de ellos como CART [34], C5.0 [35] y su versión anterior C4.5 [30] han sido utilizados en la detección de AM [36,37,38]. En este modelo, se genera una regla para cada hoja del árbol, siguiendo el camino que conduce de la raíz a la hoja.

Como se muestra en la Figura 8, el primer paso consiste en generar un árbol de decisión. El criterio utilizado para construir el árbol de decisión puede variar en dependencia de la propuesta. Luego, se ejecuta un proceso de poda sobre el árbol de decisión generado, obteniéndose una colección de árboles. Según el método aplicado, la colección obtenida puede comprender uno o varios árboles. Finalmente, tiene lugar un proceso de optimización. Dependiendo de la propuesta, el proceso de optimización puede ser aplicado sobre la colección de árboles T o sobre el conjunto de reglas R, obtenido a partir del árbol de decisión t_1, donde $t_1 \in T$. En la Tabla 2 se describe la estrategia que siguen en cada paso los métodos C5.0 y CART.

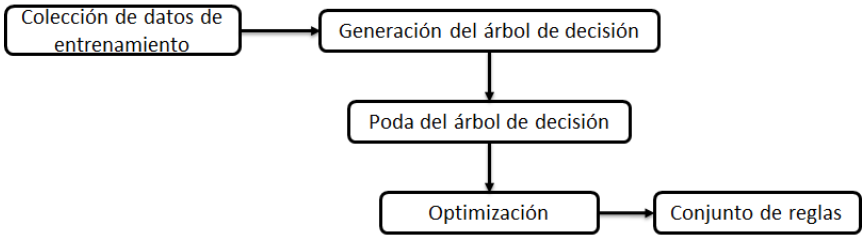

Fig. 8. Modelo de árboles de decisión.

Las reglas obtenidas por el método C5.0 pueden contener condiciones irrelevantes, por lo cual deben ser generalizadas eliminando dichas condiciones sin que esto afecte su presición. Por ejemplo, supongamos que tenemos la siguiente regla $r_1 = \langle \overleftarrow{r_1}, c \rangle$, y una regla más general $r_2 = \langle \overleftarrow{r_2}, c \rangle$, donde $\overleftarrow{r_2}$ es el resultado de eliminar una condición m del conjunto $\overleftarrow{r_1}$. El criterio para seleccionar a m puede variar de acuerdo al contexto de aplicación.

El proceso para seleccionar el árbol final en ocasiones suele ser inestable. La solución radica en utilizar la regla de un-error-estándar (en inglés one-standard-error) propuesta por Breiman *et al.* [34]. Lo que se busca con aplicar esta regla es reducir la inestabilidad al escoger el árbol final y encontrar el subárbol podado más simple cuyo rendimiento sea comparable con el del subárbol de máximo rendimiento hasta el momento. Luego de seleccionar el árbol final se genera el conjunto

Tabla 2. Resumen de las estrategias utilizadas en cada paso del modelo de árboles de decisión por los métodos C5.0 y CART.

	C5.0	CART
Generación del árbol de decisión	Se utiliza el criterio de la máxima ganancia de información para determinar cual condición debe estar en determinado nodo de decisión [39].	Se utiliza el criterio de la ganancia Gini para evaluar cual división en un nodo resulta más óptima para particionar la colección de entrenamiento [40].
Poda del árbol de decisión	El método de poda comienza por el final del árbol y examina cada subárbol que no sea hoja. Si reemplazar el subárbol en análisis por una de sus hojas o por su rama utilizada con mayor frecuencia reduce el porcentaje de error estimado, entonces se poda el árbol efectuando el reemplazo.	El valor de costo-complejidad se calcula varias veces durante la poda, como resultado de sustituir cada arista del árbol por el nodo raíz de dicha arista. El menor valor de costo-complejidad obtenido en este proceso va a indicar por donde debe ser podado el árbol. Este proceso se repite recursivamente hasta que el subárbol que se vaya a podar solo tenga el nodo raíz. Cada vez que se ejecuta el proceso de poda, el subárbol obtenido se adiciona al a colección de subárboles.
Optimización	Se extraen las reglas del árbol de decisión y se optimizan mediante un proceso de generalización.	De la colección de subárboles obtenidos durante el proceso de poda, se selecciona el subárbol más óptimo aplicando la validación cruzada. A partir del subárbol seleccionado, se extraen las reglas.

de reglas. Para ello se toma el camino que conduce de la raíz a cada hoja como una regla y se agrega al conjunto de estas.

3.5. Modelo incremental

El proceso de aprendizaje de un método se considera incremental, si las instancias de entrenamiento están disponibles durante el transcurso del tiempo, de tal manera que cada nueva instancia obliga a modificar las reglas existentes para que clasifiquen correctamente la nueva instancia [41]. Existen algunas propuestas basadas en el modelo incremental como son FLORA [42], AQ11-PM+WAH [43], FACIL [44], VFDR [45] y RILL [46].

Como se muestra en la Figura 9, cuando una nueva instancia n se encuentra disponible, esta pasa por una prueba de cobertura con el objetivo de encontrar cuales reglas la cubren. Este paso va a proveer la información necesaria para determinar si se debe generalizar una regla existente o crear una nueva en la etapa de inducción de reglas. Una vez concluida dicha etapa, se actualizan los conjuntos de instancias y de reglas. Por último, se evalúa el conjunto de reglas R actualizado sobre nuevas instancias para detectar AM.

El método FLORA utiliza tres conjuntos de reglas llamados: (1) descriptores aceptados (ADES por sus siglas en inglés), (2) descriptores negativos (NDES por sus siglas en inglés) y (3) descriptores potenciales (PDES por sus siglas en inglés). ADES contiene las reglas que cubren solamente las instancias positivas, NDES comprende aquellas reglas que cubren las instancias negativas y en PDES están las reglas que cubren tanto instancias negativas como positivas.

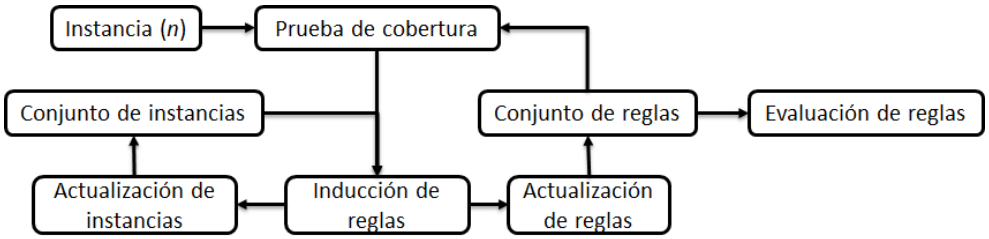

Fig. 9. Modelo incremental.

Por otra parte, el método AQ11-PM+WAH almacena instancias extremas en el conjunto de instancias. Durante la etapa de actualización de instancias, el conjunto de reglas es modificado para que cubra las instancias que se encuentran en los límites de las reglas. Para cada regla se buscan los valores mínimo y máximo de cada atributo. Luego, cada condición de la regla se modifica para crear un intervalo entre los valores mínimo y máximo. Aquellas instancias de la colección de entrenamiento D que utilizando una estrategia de cotejo exacto coinciden con los límites de la regla transformada son definidas como instancias extremas.

FACIL representa otro método basado en el modelo incremental. En este método, la generalización necesaria para que una regla r pueda describir una nueva instancia n viene dada por la medida de crecimiento ϱ [41]. El valor de ϱ se calcula para cada regla con la misma clase c_l que la instancia n. Si una regla r tiene el valor mínimo de ϱ y n se puede cubrir con un crecimiento moderado de r, entonces r se convierte en un candidato. En el caso en que una regla r_1 con diferente c_l cubra a n, el valor de su soporte negativo aumenta y n se adiciona al conjunto de instancias de r_1.

El método VFDR emplea una estructura de datos que contiene información para la clasificación de nuevas instancias y para extender las reglas. Cada regla r tiene asociada una estructura de datos L_r. Dicha estructura es equivalente al conjunto de instancias que se almacenan en las propuestas anteriores, solo que L_r además almacena estadísticas para calcular la entropía de cada clase. Durante la etapa de inducción de reglas, si una regla r cubre una instancia n se procede a actualizar L_r. La cantidad de instancias en L_r es comparada con el límite de Hoeffding H, el cual establece el número de instancias necesario para que el conjunto de reglas se actualice, ya sea extendiendo una regla existente o induciendo una nueva. Para extender una regla, se calcula la medida de evaluación de la división para cada atributo y sus valores observados en más del 10 % de las instancias. Si el valor de la mejor medida de división es superior al de la medida sin división, entonces la regla se extiende con un nuevo selector obtenido a partir de la mejor división.

En el método RILL, cuando una nueva instancia n está disponible se incrementa su índice y n se adiciona a una ventana deslizante W (conjunto de instancias). Este método al igual que VFDR almacena información estadística de las reglas, específicamente el número de instancias positivas cubiertas en W así como la fecha y hora de su último uso. La información estadística se actualiza y almacena para cada regla que cubra a n.

Como se muestra en la Tabla 3, cada paso del modelo incremental funciona de forma diferente según el método utilizado.

Tabla 3. Resumen de las estrategias utilizadas en cada paso por diferentes métodos basados en el modelo incremental.

	FLORA	FACIL	AQ11-PM+WAH	VFDR	RILL
Prueba de cobertura	Si una nueva instancia n se adiciona a W, su respectivo R (ADES si n es positiva o NDES si n es negativa) es evaluado para determinar si existe alguna regla que cubra a n. Si no existe ninguna regla que cubra a n, se ejecuta un proceso de generalización de reglas.	Cuando una nueva instancia n está disponible, se evalúan las reglas con la misma c_l que n para encontrar una candidata. Si ninguna de estas reglas cubre a n, entonces el resto de las reglas con diferente c_l son evaluadas.	Se comprueba si una nueva instancia n está mal clasificada por las reglas existentes. Las instancias mal clasificadas se combinan con las existentes en la memoria parcial (conjunto de instancias) para formar una colección de entrenamiento.	Cuando una nueva instancia n está disponible, todas las reglas son evaluadas.	
Inducción de reglas	Si no existe ninguna regla y ninguna generalización que cubra a n, se genera una regla cuya $\overleftarrow{r}$ coincide con n y se adiciona a su respectivo R.	Si el nivel de pureza de r_1 se reduce por debajo de un umbral mínimo establecido, entonces se generan nuevas reglas a partir de las instancias asociadas a r_1 y se adicionan a R. Si r_1 no cubre a n, se calcula la intersección ι de r_1 con el candidato. Si $\iota \neq \emptyset$, entonces el candidato se descarta y se genera una regla específica r_n describiendo a n y se adiciona a R. Si $\iota = \emptyset$, el candidato es generalizado con respecto a n y se adiciona a R.	Para inducir una regla r_1, se selecciona de forma aleatoria una instancia positiva llamada semilla (en inglés seed) y se generaliza tanto como sea posible teniendo en cuenta las restricciones de las instancias negativas. Las instancias positivas cubiertas por r_1 son eliminadas de D y se repite el todo proceso hasta que todas las instancias positivas de D sean cubiertas.	Se calcula el valor de la entropía de L_r. Si la entropía excede a H, entonces r se debe extender. Si ninguna de las reglas cubre a n, las estadísticas L_{r_1} de la regla por defecto r_1 se comparan con H. Si $L_{r_1} > H$, se induce una nueva regla a partir de r_1.	Si ninguna regla positiva cubre a n, se lleva a cabo un proceso de generalización. En caso de que el proceso de generlización no obtenga una regla que cubra a n, se genera una regla con la descripción completa de n y se adiciona al conjunto de reglas como la regla más específica.
Actualización de reglas	Se evalúa el conjunto PDES y se incrementan los contadores positivos de las reglas que cubren a n. Finalmente, se evalúa el conjunto de reglas opuesto (NDES si n es positiva o ADES si n es negativa), aquellas reglas que cubran a n se mueven al conjunto PDES y se actualizan sus contadores.	Si el soporte de una regla r es menor que que el soporte de cualquier regla generada a partir de r, entonces r se elimina.	Las reglas obtenidas durante el proceso de inducción de reglas son adicionadas al conjunto de reglas.		Una regla es eliminada si: el tiempo de uso es superior a un umbral máximo establecido, tiene muy bajo nivel de pureza o tiene muchos errores de predicción.
Actualización de instancias	Se elimina la instancia más antigua de W si su capacidad está completa. Esto puede causar que una regla se elimine o se mueva de PDES a ADES o NDES, en dependencia del tipo de instancia que se eliminó (negativa o positiva).	Una instancia se elimina si: es más antigua que un umbral establecido o deja de ser relevante (ya no está en los límites de ninguna regla).	Las instancias extremas son combinadas con las obtenidas anteriormente. Las instancias se eliminan cuando ya no coinciden con los límites de las reglas.	Se actualiza L_r cuando r cubre una instancia etiquetada.	Se elimina la instancia más antigua si la cantidad de instancias almacenadas en W excede su umbral máximo.

4. Discusión

Luego de analizar las propuestas presentadas en este trabajo, fueron detectadas algunas limitaciones en términos de efectividad cuando son aplicadas en escenarios como los descritos anteriormente (ver Sección 1).

En los métodos basados en el modelo de árboles de decisión, el preprocesamiento de datos dedicado exclusivamente a detectar datos faltantes puede conllevar a que el árbol obtenido alcance un elevado nivel de complejidad. Esto trae consigo que se genere una gran cantidad de reglas, muchas de estas con condiciones irrelevantes, lo cual afecta su rendimiento mediante el aumento del consumo de memoria y el tiempo de procesamiento.

También existen limitaciones en cuanto al tipo de atributo que pueden procesar. Por ejemplo, algunos métodos basados en el modelo de búsqueda voraz sólo pueden procesar atributos categóricos, y para poder procesar atributos continuos requieren de un proceso de discretización. Esto causa que los métodos se vean limitados en cuanto a los operadores que pueden utilizar en las reglas que generen; obligándolos a sólo poder emplear operadores tales como "=" o "≠".

Tanto el modelo de árboles de decisión como el modelo de búsqueda voraz están diseñados para entornos estáticos, lo cual conlleva a que no se contemple el cambio de concepto que puede tener lugar en el flujo de datos. Este hecho dificulta la detección de actividades maliciosas que hayan sido modificadas para evadir el mecanismo de detección. Además, las reglas existentes no evolucionan de forma automática, impidiendo que puedan adaptarse a posibles cambios en los escenarios.

Los métodos basados en el modelo incremental presentados en este trabajo, no han sido evaluados sobre grandes volúmenes de datos. No obstante, una de sus características es que almacenan instancias después del proceso de aprendizaje. Teniendo en cuenta las características de los escenarios descritos anteriormente (ver Sección 1), su rendimiento puede verse afectado debido a la alta cantidad de instancias y atributos que estas pueden tener. Además, el modelo incremental carece de una etapa de preprocesamiento de datos, lo cual pudiera afectar la cantidad y la complejidad de las reglas generadas.

Resumiendo, en el modelo general basado en reglas de clasificación para la detección de AM presentado anteriormente fueron detectados seis problemas. Dos de ellos relacionados con la etapa de preprocesamiento de datos y cuatro relacionados con la etapa de generación reglas.

Problemas en la etapa de preprocesamiento de datos:

- En algunos casos no se realiza, ignorándose la dimensionalidad de los datos y el alto número de instancias. (Afecta la eficiencia del modelo.)
- En los casos donde se aplica el preprocesamiento de datos no se tiene en cuenta la elevada cantidad de instancias en la colección de entrenamiento. (Afecta la eficiencia del modelo.)

Problemas en la etapa de generación de reglas:

- Alto número de reglas. (Afecta la eficiencia del modelo y la QoS.)
- Reglas con condiciones irrelevantes. (Afecta la efectividad del modelo.)
- Inconsistencia entre las reglas en algunos casos. (Afecta la efectividad del modelo y la QoS.)
- En algunos casos las reglas existentes no pueden modificarse de forma automática. (Afecta la efectividad del modelo.)

5. Conclusiones

La detección de actividades maliciosas recientemente se convirtió en un tema popular de investigación. Los modelos basados en reglas son reportados como uno de los más utilizados para detectar en el menor tiempo posible eventos asociados a actividades maliciosas. En este trabajo, se analizaron varios modelos basados en reglas para detectar actividades maliciosas. Con el objetivo de lograr una mayor efectividad en la detección de actividades maliciosas, tales modelos deben ser capaces de manejar problemas relacionados con la dimensionalidad de los datos y el cambio de concepto. Además, un modelo de este tipo no debería reducir el nivel de QoS definido por la entidad donde será desplegado.

Sería interesante realizar una comparación entre los métodos presentados sobre diferentes colecciones de datos. Lamentablemente, sus implementaciones no están disponibles de forma pública. Además, algunas de las colecciones de datos utilizadas en los trabajos presentados no son públicas, lo cual se debe a razones de privacidad y limitaciones legales. Este hecho hace que sea difícil establecer una comparación entre los resultados obtenidos por diferentes métodos.

Algunos de los problemas identificados durante el análisis de los modelos, indican que tanto su efectividad como la calidad del servicio pueden ser afectados. Esos problemas pueden ser abordados en futuras investigaciones para proponer nuevas soluciones diseñadas para hacer la detección de actividades maliciosas una tarea más efectiva.

Referencias bibliográficas

1. CFCA: 2013 global fraud loss survey. [citado 20 de abril de 2016]. Disponible en Internet: http://www.cvidya.com/media/62059/global-fraud_loss_survey2013.pdf (2013)
2. Dua, S., Du, X.: Data mining and machine learning in cybersecurity. CRC press (2016)
3. Abdallah, A., Maarof, M.A., Zainal, A.: Fraud detection system: A survey. Journal of Network and Computer Applications **68** (2016) 90–113
4. Herrera-Semenets, V., Prado-Romero, M.A., Gago-Alonso, A.: Análisis de los métodos de detección de fraude en servicios de telecomunicaciones. Technical Report RT_023, Serie Gris, Advanced Technologies Application Center (CENATAV), La Habana, Cuba (February 2014)
5. Anika, N., Sharmistha, R., Syeda, S.H.: A survey on different approaches used for credit card fraud detection. International Journal of Applied Information Systems **10**(4) (2016) 29–34
6. Shukran, M.A.M., Maskat, K.: An intelligent network intrusion detection using data mining techniques. Jurnal Teknologi **76**(12) (2015)
7. Tesfahun, A., Bhaskari, D.L.: Effective hybrid intrusion detection system: A layered approach. International Journal of Computer Network and Information Security (IJCNIS) **7**(3) (2015) 35
8. Rivero-Pérez, J.L.: Técnicas de aprendizaje automático para la detección de intrusos en redes de computadoras. RCCI **8**(4) (2014) 37–49
9. SAP: Detect and prevent fraud to reduce financial loss. [Online]. Available: http://www.sap.com/bin/sapcom/de_de/downloadasset.2013-09-sep-17-10.detect-prevent-and-deter-fraud-in-big-data-environments-pdf.html (2013)
10. Gurav, S.S., Todmal, S.R.: A survey on activity detection using data mining. International Journal of Innovative Research in Computer and Communication Engineering **2**(11) (2014) 6947–6952
11. de Gouveia, F., Magedanz, T.: Quality of service in telecommunication networks. telecommunication system and technologies **2** (2002)
12. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery in databases. AI magazine **17**(3) (1996) 37–54
13. Azevedo, A.I.R.L.: Kdd, semma and crisp-dm: a parallel overview. IADS-DM (2008)
14. Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R.: CRISP-DM 1.0: Step-by-step data mining guide. SPSS Inc. (2000)

15. Kdnuggets: Crisp-dm, still the top methodology for analytics, data mining, or data science projects. [citado 10 de junio de 2016]. Disponible en Internet: http://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html
16. Ahn, S.H., Kim, N.U., Chung, T.M.: Big data analysis system concept for detecting unknown attacks. In: Advanced Communication Technology (ICACT), 2014 16th International Conference on, IEEE (2014) 269–272
17. Wirth, R., Hipp, J.: CRISP-DM: Towards a standard process model for data mining. In: Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining, Citeseer (2000) 29–39
18. Fanaee-T, H., Gama, J.: Event labeling combining ensemble detectors and background knowledge. Progress in Artificial Intelligence **2**(2-3) (2014) 113–127
19. Herrera-Semenets, V., Gago-Alonso, A.: Búsqueda automática de reglas para detección de fraudes en flujos de eventos. Technical Report RT_029, Serie Gris, Advanced Technologies Application Center (CENATAV), La Habana, Cuba (January 2015)
20. Han, J., Kamber, M., Pei, J.: Data mining: concepts and techniques. Elsevier (2011)
21. SUPENO, D., BASKORO, A.P., HUDAN, S., RADITYO, A., HENNING, T.: Coro: Graph-based automatic intrusion detection system signature generator for evoting protection. Journal of Theoretical and Applied Information Technology **81**(3) (2015) 535–546
22. Mokube, I., Adams, M.: Honeypots: concepts, approaches, and challenges. In: Proceedings of the 45th annual southeast regional conference, ACM (2007) 321–326
23. Michalski, R.S., Kaufman, K.A., Pietrzykowski, J., Wojtusiak, J., Mitchell, S., Seeman, D.: Natural induction and conceptual clustering: A review of applications. Reports of the Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA, USA (2006)
24. Liu, H., Tan, S.T.: X2R: A fast rule generator. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, Vancouver, Canada (October 1995)
25. Cohen, W.W.: Fast effective rule induction. In Proceedings of the 12th international conference on machine learning, Tahoe City (1995) 115–123
26. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. University of Waikato, Department of Computer Science (1998)
27. Parpinelli, R.S., Lopes, H.S., Freitas, A.A.: An ant colony algorithm for classification rule discovery. Data Mining: A Heuristic Approach **208** (2002) 191–208
28. Quinlan, J.R.: MDL and categorical theories (continued). In: In Machine Learning: Proceedings of the Twelfth International Conference, Lake Taho, Morgan Kaufmann (1995) 464–470
29. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on **26**(1) (1996) 29–41
30. Quinlan, J.R.: C4.5: programs for machine learning. San Francisco, California, Morgan Kaufmann (1993) 302 p.
31. Barthakur, P., Dahal, M., Ghose, M.K.: Adoption of a fuzzy based classification model for p2p botnet detection. International Journal of Network Security **17**(5) (2015) 522–534
32. Hoque, N., Bhattacharyya, D.K., Kalita, J.K.: Botnet in ddos attacks: trends and challenges. IEEE Communications Surveys & Tutorials **17**(4) (2015) 2242–2270
33. Hühn, J., Hüllermeier, E.: FURIA: an algorithm for unordered fuzzy rule induction. Data Mining and Knowledge Discovery, Springer **19** (2009) 293–319
34. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and regression trees. CRC press (1984)
35. Kuhn, M., Johnson, K.: Applied predictive modeling. Springer (2013)
36. Pinem, A.F.A., Setiawan, E.B.: Implementation of classification and regression tree (cart) and fuzzy logic algorithm for intrusion detection system. In: 2015 3rd International Conference on Information and Communication Technology (ICoICT), IEEE (2015) 266–271
37. Nia, F.Y., Khalili, M.: An efficient modeling algorithm for intrusion detection systems using c5. 0 and bayesian network structures. In: 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI), IEEE (2015) 1117–1123
38. Kevric, J., Jukic, S., Subasi, A.: An effective combining classifier approach using tree algorithms for network intrusion detection. Neural Computing and Applications (2016) 1–8
39. Xiaoliang, Z., Jian, W., Hongcan, Y., Shangzhuo, W.: Research and application of the improved algorithm c4. 5 on decision tree. In: Test and Measurement, 2009. ICTM'09. International Conference on. Volume 2., IEEE (2009) 184–187
40. Lawrence, R.L., Wright, A.: Rule-based classification systems using classification and regression tree (CART) analysis. Photogrammetric engineering and remote sensing **67**(10) (2001) 1137–1142

41. Deckert, M.: Incremental rule-based learners for handling concept drift: an overview. Foundations of Computing and Decision Sciences **38**(1) (2013) 35–65
42. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine learning **23**(1) (1996) 69–101
43. Maloof, M., et al.: Incremental rule learning with partial instance memory for changing concepts. In: Neural Networks, 2003. Proceedings of the International Joint Conference on. Volume 4., IEEE (2003) 2764–2769
44. Ferrer-Troyano, F.J., Aguilar-Ruiz, J.S., Santos, J.C.R.: Incremental rule learning and border examples selection from numerical data streams. J. UCS **11**(8) (2005) 1426–1439
45. Gama, J., Kosina, P., et al.: Learning decision rules from data streams. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence. Volume 22., Citeseer (2011) 1255–1260
46. Deckert, M., Stefanowski, J.: Rill: Algorithm for learning rules from streaming data with concept drift. In: Foundations of Intelligent Systems. Springer (2014) 20–29

RT_035, julio 2016
Aprobado por el Consejo Científico CENATAV
Derechos Reservados © CENATAV 2016
Editor: Lic. Lucía González Bayona
Diseño de Portada: Di. Alejandro Pérez Abraham
RNPS No. 2143
ISSN 2072-6260
Indicaciones para los Autores:
Seguir la plantilla que aparece en www.cenatav.co.cu
C E N A T A V
7ma. A No. 21406 e/214 y 216, Rpto. Siboney, Playa;
La Habana. Cuba. C.P. 12200
Impreso en Cuba