UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Tecnura

http://revistas.udistrital.edu.co/ojs/index.php/Tecnura/issue/view/650

DOI: http://dx.doi.org/10.14483/udistrital.jour.tecnura.2015.2.a08

INVESTIGACIÓN

# Extracción de reglas de clasificación sobre repositorio de incidentes de seguridad informática mediante programación genética

## Extracting classification rules from an informatic security incidents repository by genetic programming

Carlos Javier Carvajal Montealegre*

**Fecha de recepción:** 21 de enero de 2014 **Fecha de aceptación:** 19 de enero de 2015

**Citation / Para citar este artículo:** Carvajal Montealegre, C. J. (2015). Extracción de reglas de clasificación sobre repositorio de incidentes de seguridad informática mediante programación genética. *Revista Tecnura, 19*(44), 109-119. doi:http://dx.doi.org/10.14483/udistrital.jour.tecnura.2015.2.a10

### Resumen

En este artículo se describe la obtención de reglas de clasificación sobre una colección de datos de incidentes de seguridad informática en un proceso de minería de datos, detallando el uso de la programación genética como un medio para modelar el comportamiento de los incidentes y representar las reglas en árboles de decisión. El proceso de extracción descrito incluye varios puntos, como la evaluación del enfoque de programación genética, la forma de representar a los individuos y la afinación de los parámetros del algoritmo para elevar el rendimiento. Se concluye con un análisis de los resultados y la descripción de las reglas obtenidas, considerando las posibles soluciones para minimizar la ocurrencia de los ataques informáticos. El artículo se basa en una parte de la tesis de grado Análisis de Incidentes de Seguridad Informática Mediante Minería de Datos, para Modelado de Comportamiento y Reconocimiento de Patrones (Carvajal, 2012).

**Palabras clave:** áboles de decisión, colección de datos, minería de datos, programación genética, seguridad informática.

### Abstract

This paper describes the data mining process to obtain classification rules over an information security incident data collection, explaining in detail the use of genetic programming as a mean to model the incidents behavior and representing such rules as decision trees. The described mining process includes several tasks, such as the GP (Genetic Programming) approach evaluation, the individual's representation and the algorithm parameters tuning to upgrade the performance. The paper concludes with the result analysis and the description of the rules obtained, suggesting measures to avoid the occurrence of new informatics attacks. This paper is a part of the thesis work degree: Information Security Incident Analytics by Data Mining for Behavioral Modeling and Pattern Recognition (Carvajal, 2012).

**Keywords:** data collection, data mining, decision trees, genetic programming, information security.

* Ingeniero de Sistemas, Oracle Certified Associate Java SE 7 Programmer, Oracle Certified Professional Java SE 7 Programmer, Oracle Certified Expert Java EE 6 Web Services Developer. Chief Technology Officer en Conectar.biz. Bogotá, Colombia. Contacto: ing.carlosj@gmail.com

## INTRODUCCIÓN

La seguridad informática es hoy día un asunto de importancia mundial, dado el grado de conectividad logrado gracias a Internet, que ha incluido la tecnología en aéreas que hace algunos años parecían impensables, como la interacción social, que ahora es posible gracias a redes como Facebook, Twitter, Google+ e Instagram. De igual manera los entornos empresariales, financieros y académicos (entre otros), coexisten en el mundo de la Web 2.0. A su vez, los ataques informáticos aumentan también su ocurrencia (Software Engineering Institute Carnagie Mellon, 2010) y gravedad, vulnerando sectores más sensibles relacionados con la vida diaria. Lo anterior reviste de relevancia los análisis posibles sobre la información de estos incidentes de seguridad informática. Por estas razones, se realizó una extracción de reglas de clasificación para incidentes de seguridad informática.

En la minería de datos una de las tareas más comunes es la clasificación, en la cual se elige un atributo del conjunto de datos que será denominado *clase*; la pertenencia de un registro de dicho conjunto a una clase depende de los demás atributos que lo conformen. Existen varios métodos para efectuar esta clasificación, como los árboles de decisión, clasificación bayesiana, redes neuronales y vecinos próximos (Han, 2005); cada uno de estos métodos permite obtener un modelo de clasificación, el cual recibe un nuevo registro sin valor de clase y estima dicho valor. Estos modelos no son siempre interpretables, pero en el caso de los árboles de decisión el modelo se puede traducir a *reglas de clasificación*.

Las reglas de clasificación se representan en forma de sentencias de tipo IF-THEN, establecen la pertenencia de un nuevo registro a una clase de acuerdo con los valores de sus atributos; estos valores de atributo se conocen como antecedentes y la presencia de estos implica la predicción de la clase, denominada consecuente. Por ejemplo, If $X_1$ and $X_2$ and ... $X_n$ Then Y, donde $X_i$, $\forall i \in \{1,2,...,n\}$ es un antecedente que lleva a la predicción de la clase Y. Las reglas de clasificación permiten construir un modelo de clasificación de fácil interpretación para modelar el comportamiento del conjunto de datos.

La *programación genética* (Poli, 2008) se presenta como una alternativa para la extracción de reglas (Luna, 2012) (Mendes, 2001), "hereda" las ventajas de los algoritmos genéticos en cuanto a la rapidez de exploración del espacio de soluciones y elimina la representación estática de los individuos, los cuales pueden tener tamaño y formas variables (como las reglas de clasificación).

## METODOLOGÍA

A continuación se detallan las etapas realizadas para la extracción de reglas.

Etapa I. Análisis y evaluación de alternativas: en esta etapa se hizo una evaluación de las alternativas para la extracción de reglas y se eligieron el enfoque y herramientas para el proceso que se va a realizar.

Etapa II. Experimentación: en esta etapa se hizo una preparación de datos para la ejecución de los experimentos, se dio un ajuste inicial a los parámetros experimentales y se ejecutó la extracción midiendo el rendimiento de cada ejecución.

Etapa III. Análisis de resultados: con las mediciones de cada experimento, se tomaron las reglas obtenidas de las ejecuciones con mejores resultados, se tradujeron dichas reglas a un árbol de decisión que permitió interpretarlas más fácilmente y se concluyó con el comportamiento presentado según estas reglas de clasificación para los incidentes de seguridad.

## PROGRAMACIÓN GENÉTICA

La programación genética (PG) difiere de los algoritmos genéticos (AGs) tradicionales en la forma en que los individuos son representados, siendo común en AGs que los individuos sean un arreglo de tamaño fijo en donde cada posición puede contener valores numéricos o alfanuméricos. Para codificar las características de un individuo de la población, se da un valor a cada posición del arreglo, esta representación es adecuada para los problemas de

optimización (principal uso de los AGs); no obstante, los AGs no permiten individuos con tamaños y formas dinámicas, lo que limita su campo de acción (Wong, 2000) en cuanto a la búsqueda de reglas de clasificación y a la simulación de programas de computación permitido por la PG.

El concepto de evolución biológica (reproducción, selección, supervivencia del más fuerte) es de utilidad en la solución de otros problemas distintos a los de optimización, como lo es la extracción de reglas (Banzhaf, 1998), por lo que se extiende este paradigma a la programación genética.

Para la PG, los individuos tienen estructuras similares a los programas de computador con jerarquía, de tamaño y forma variables, al igual que los AGs se realizan las fases de:

- Generación de una población inicial.
- Asignación de un valor de *fitness* para cada individuo. El *fitness* se asigna mediante una función, también conocida como función de evaluación o función objetivo. El *fitness* es una medida de desempeño del individuo y depende enteramente del contexto del problema evaluado.
- Creación de una nueva población de individuos a partir de la copia de los ya existentes o mediante la recombinación genética y mutación.

La PG hace la exploración en un espacio de soluciones compuesto por programas (o algoritmos) de computador que pretenden dar solución a un problema, los individuos se representan como árboles (Han, 2005), definidos por R. Koza como "el conjunto de las posibles composiciones de funciones que puedan ser creadas recursivamente del conjunto de $N_{func}$ funciones de $F=\{f_1,f_2,...,f_{Nfunc}\}$ y el conjunto de $N_{term}$ terminales de $T=\{a_1,a_2,...,a_{Nterm}\}$" (Koza,1992). Las funciones pueden ser:

- Operadores aritméticos (+,-,*,/).
- Funciones matemáticas (seno, coseno, exponenciación, logaritmo, valor absoluto).
- Operadores de lógica booleana (AND, OR, NOT).
- Operadores condicionales (If-Then-Else).
- Iteraciones (Do-Until).
- Funciones recursivas.

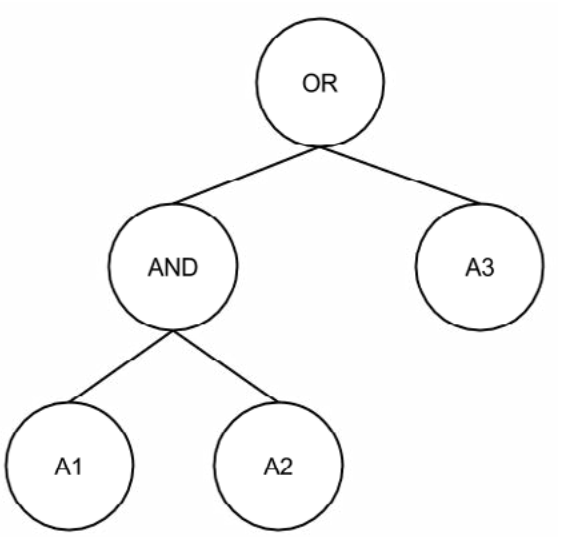


**Figura 1.** Ejemplo de una regla lógica representada en forma de árbol.

**Fuente:** (Carvajal, 2012).

Las terminales son variables o constantes atómicas cuyos valores representan alguna característica particular del problema. En la figura 1 se puede observar un ejemplo de un árbol de decisión.

El cruce de los individuos en busca del mejor espécimen se define para árboles como la elección de un nodo al azar de los dos padres, para realizar un intercambio de sub-árboles; este operador de reproducción está fuertemente ligado al concepto de *clausura,* sin el cual un cruce de dos árboles podría generar individuos inviables (Bojarczuk, 2004).

Para la mutación de los árboles, primero se reconoce el tipo de nodo que será modificado; si pertenece al conjunto de terminales T, basta con tomar otra terminal del conjunto para remplazar el valor del nodo actual; si el nodo es una función perteneciente al conjunto F, se tomará otra función del conjunto asegurándose de mantener la *clausura*.

### Obtención de reglas de calificación mediante programación genética

Usando la estructura dinámica de los individuos de la PG, es posible representar las reglas de clasificación para la evolución de las mismas y hallar el mejor individuo entre un conjunto de reglas que estimen un valor de clase. No obstante, los problemas de clasificación cuentan con muchos valores de clase, lo que implica un número igual o mayor de reglas, así que si obtenemos como resultado final de la evolución a la mejor regla, siendo esta solo válida para un valor de clase, la solución del problema estaría incompleta. La PG trata este inconveniente mediante dos aproximaciones (Freitas, 2002):

- *Michigan:* en donde cada individuo es una regla de clasificación, así que la solución al problema se compone de un subconjunto de la población o la totalidad de la misma.
- *Pittsburgh:* cada individuo es un conjunto de reglas de clasificación, representando una posible solución del problema.

## EXTRACCIÓN DE REGLAS SOBRE UN REPOSITORIO DE DATOS CATEGÓRICOS

Dada la utilidad de la programación genética, se efectuó un proceso de extracción de reglas de clasificación sobre un conjunto de datos de incidentes de seguridad informática, con el fin de obtener un modelo de comportamiento de dichos incidentes de seguridad.

### Descripción del repositorio

El conjunto de datos se compone de 1234 registros de incidentes de seguridad informática recopilados alrededor de todo el mundo desde el año 2004 hasta 2011, obtenidos de 3 fuentes principales: Web Hacking Incident DataBase (The Web Application Security Consortium, 2010), Chronology of Data Breaches (Privacy Rights Clearing House, 2010) y COL-CSIRT (Universidad Distrital Francisco José de Caldas, 2010). Los campos del repositorio se describen en la tabla 1. El campo MÉTODO DE ATAQUE fue elegido como valor de clase.

**Tabla 1.** Campos del repositorio de incidentes de seguridad informática.

| Campo | Descripción |
|---|---|
| Tipo de organización | Actividad desempeñada por la entidad atacada. |
| Entidad | Nombre de la entidad víctima del incidente de seguridad informática. |
| Método de ataque | Método usado por el atacante para causar el incidente de seguridad informática. |
| Debilidad de la aplicación | Falencia que permitió que el incidente de seguridad fuera posible. |
| País | País donde se ubica la entidad que fue víctima del incidente de seguridad informática. |
| Fecha | Fecha en la cual ocurrió el incidente de seguridad informática. |
| Atacante | Criminales profesionales, espías, hackers, intrusos corporativos, terroristas, usuario externo, usuario interno, vándalos. |
| Herramienta | Agente autónomo, ataque físico, comando de usuario, intercambio de información, no aplica, script o programa, toolkit. |
| Vulnerabilidad | Configuración, diseño, implementación, políticas de seguridad. |
| Blanco | Componente, computador, cuenta, dato, proceso, red. |
| Resultado no autorizado | Acceso incrementado, corrupción de la información, denegación del servicio, difusión de la información, robo de recursos. |
| Objetivo | Cambio de estatus, daño, ganancia financiera, ganancia política, sin intencionalidad. |

**Fuente:** (Carvajal, 2012).

## Proceso de extracción de reglas

Para la ejecución de los experimentos se usó un *framework* desarrollado totalmente en JAVA y de uso libre JCLEC (Knowledge Discovery and Intelligent Systems, 2011); este framework tiene implementadas las aproximaciones propuestas por C. Bojarczuk (Bojarczuk, 2004), I. De Falco (De Falco, 2001) y K. C. Tan (Tan, 2002); se eligió finalmente la propuesta de K.C. Tan, pues además de implementar todos los operadores lógicos y matemáticos básicos permite la mutación e incluye el concepto de torneo de tokens (Wong, 2000) para optimizar la evolución mediante la penalización del fitness. En el contexto de extracción de reglas, el fitness se asigna según la precisión de la regla obtenida al clasificar los registros en los conjuntos de entrenamiento y de evaluación. En la tabla 2 se resumen las características de los tres enfoques evaluados.

El algoritmo admite como parámetros los porcentajes de mutación, cruce, reproducción, tamaños de los conjuntos de entrenamiento y prueba, tamaño de la población, número máximo de generaciones y los valores de las variables $W_1$ y $W_2$, las cuales se emplean para determinar el fitness de los individuos, el cual se calcula como en la ecuación (1).

$$Fitness = \frac{tp}{(tp + w_1 fn)} * \frac{tn}{(tn + w_2 fp)} \quad (1)$$

tp: Verdaderos positivos (true positives),
fn: Falsos negativos (false negatives),
tn: Verdaderos negativos (true negatives),
fp: Falsos positivos (false positives).

Por otra parte, el torneo de tokens modifica el fitness de los individuos penalizándolos como se muestra en la ecuación (2).

**Tabla 2.** Características de los tres enfoques evaluados para extracción de reglas.

| | C. Bojarczuk | I. De Falco | K. C. Tan |
|---|---|---|---|
| Copiado | X | X | X |
| Mutación | | X | X |
| Recombinación | X | X | X |
| Representación | Pittsburgh/Michigan | Michigan | Michigan |
| OPERADORES | | | |
| AND | X | X | X |
| OR | X | X | X |
| NOT | | X | X |
| < | | X | X |
| <= | X | X | X |
| > | X | X | X |
| >= | | X | X |
| = | X | X | X |
| != | X | X | X |
| IN | | X | X |
| OUT | | X | X |
| FITNESS | | | |
| Sensibilidad | X | X | X |
| Especificidad | X | | X |
| Simplicidad | X | X | |

**Fuente:** (Carvajal, 2012)

$$Fitness\ Ajustado = fitness * \frac{conteo\_tokens}{conteo\_ideal} \quad (2)$$

Donde el conteo_tokens indica cuántos registros fueron clasificados correctamente por la regla y el conteo_inicial es el total de registros que debieron ser clasificados por la regla.

Para la ejecución de los experimentos se usó 70% de los datos para el entrenamiento (864 registros) y 30% para las pruebas (370 registros), el tamaño de la población se modificó para 100,300 y 600 individuos, mientras que para el número de generaciones se usaron valores de 100 y 200.

El valor de reproducción (copiado) se mantuvo en 0,01, mientras que los valores de mutación y cruce se modificaron en un valor de 0,1 por cada ejecución aumentando uno de los operadores al tiempo que se disminuía el otro (0,1 y 0,9 para mutación y cruce respectivamente, luego 0,2 y 0,8 hasta llegar a 0,9 y 0,1). Los valores de los pesos $W_1$ y $W_2$ se modificaron de igual forma con un delta de 0,1 en cada ejecución, aumentando uno y disminuyendo el otro.

Los registros, pertenecientes a clases como Bot, Clickjacking, Comando del sistema operativo, Gusano, Hijacking DNS, Inclusión de Archivo Local, Inclusión de Archivo Remoto, Navegación Forzada, Redirection, fueron retirados del conjunto de prueba ya que tenían en promedio menos de 10 registros por clase, lo que minimiza la posibilidad de encontrar una regla satisfactoria mediante el algoritmo. Esto se visualizó en las primeras ejecuciones del algoritmo en donde aún no se retiraban estos registros y en donde la precisión de las reglas de clasificación para estas clases no superaba en el mejor de los casos 16%, y con este ajuste el número de clases posibles se redujo a 21.

Bajo estas condiciones se realizaron 18 ejecuciones, se tomaron como medidas de evaluación del desempeño de cada experimento el tiempo de ejecución, la precisión del modelo obtenida durante las etapas de entrenamiento y de prueba, el índice Cohen Kappa y el AUC (Area Under Curve).

Todas los experimentos se ejecutaron en una máquina con Intel Core i7 con 6 GB de RAM y Windows 7 como sistema operativo.

## ANÁLISIS DE RESULTADOS

Los valores de los pesos $W_1$ y $W_2$ flexibilizan la rigurosidad con que se asigna el fitness de cada individuo según la precisión de clasificación, como se vio en la ecuación (1), durante la variación de estos pesos se observó que valores bajos de $W_1$ y valores altos de $W_2$ conducen a un sobreentrenamiento del modelo (*overfitting*), mientras que valores altos de $W_1$ y bajos de $W_2$ reducen drásticamente la precisión del modelo, como se puede

**Tabla 3.** Valores de precisión durante la variación de los pesos.

| # EJECUCIÓN | PESOS | PRECISIÓN ENTRENAMIENTO | PRECISIÓN PRUEBAS |
|---|---|---|---|
| 1 | $W_1$=0,1, $W_2$=0,9 | 0,8213 | 0,687 |
| 2 | $W_1$=0,2, $W_2$=0,8 | 0,7816 | 0,6812 |
| 3 | $W_1$=0,3, $W_2$=0,7 | 0,7829 | 0,6754 |
| 4 | $W_1$=0,4, $W_2$=0,6 | 0,763 | 0,6754 |
| 5 | $W_1$=0,5, $W_2$=0,5 | 0,7717 | 0,6986 |
| 6 | $W_1$=0,4, $W_2$=0,6 | 0,7543 | 0,6551 |
| 7 | $W_1$=0,3, $W_2$=0,7 | 0,7543 | 0,6551 |
| 8 | $W_1$=0,2, $W_2$=0,8 | 0,6526 | 0,5913 |
| 9 | $W_1$=0,1, $W_2$=0,9 | 0,5993 | 0,5072 |

**Fuente:** (Carvajal, 2012).

observar en la tabla 3 y figura 2, donde los valores de mutación y cruce se mantuvieron en 0,8 y 0,1, respectivamente, con una población de 600 individuos y 200 generaciones. Cuando los pesos se ajustaron a 0,5 cada uno la precisión obtenida en entrenamiento y pruebas fue de 0,7717 y 0,6986 en comparación con los demás resultados.

Al variar los valores de los operadores de mutación y cruce manteniendo los pesos en 0,5 cada uno, se observó que los valores que mejor resultado arrojaban eran 0,6 para el cruce y 0,4 para la mutación. La precisión del modelo se observa en la tabla 4 y el comportamiento de las curvas, en la figura 3.

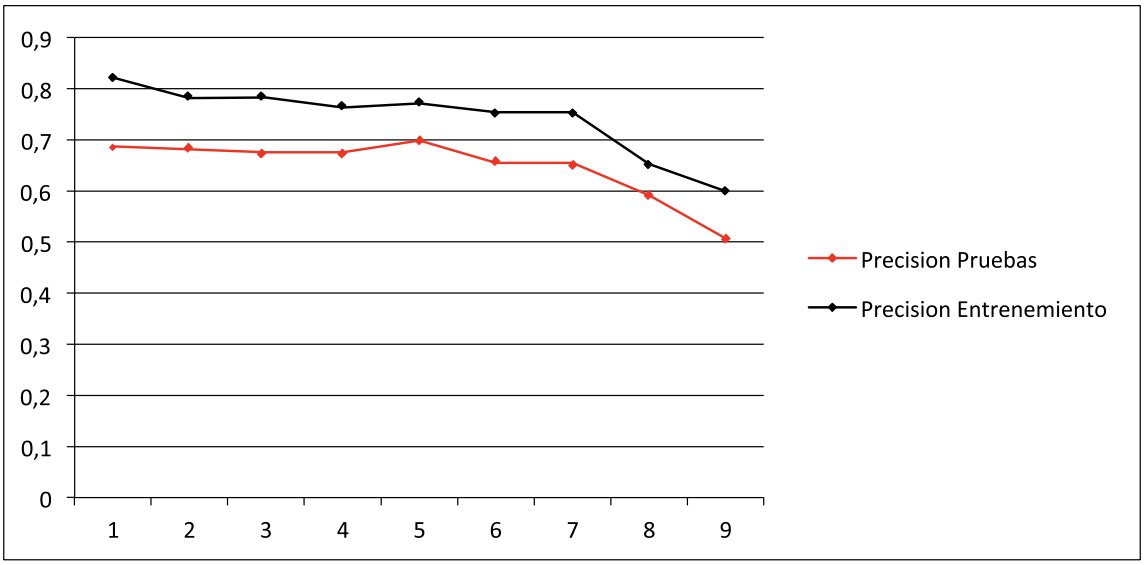


**Figura 2.** Gráfica de las curvas de precisión de pruebas vs. entrenamiento durante la variación de pesos.

**Fuente:** (Carvajal, 2012).

**Tabla 4.** Valores de precisión durante la variación de los operadores de reproducción.

| # EJECUCIÓN | CRUCE | MUTACIÓN | PRECISIÓN ENTRENAMIENTO | PRECISIÓN PRUEBAS |
|---|---|---|---|---|
| 10 | 0,9 | 0,1 | 0,7878 | 0,687 |
| 11 | 0,8 | 0,2 | 0,768 | 0,6551 |
| 12 | 0,7 | 0,3 | 0,7531 | 0,6638 |
| 13 | 0,6 | 0,4 | 0,7928 | 0,7246 |
| 14 | 0,5 | 0,5 | 0,7792 | 0,6812 |
| 15 | 0,4 | 0,6 | 0,7667 | 0,687 |
| 16 | 0,3 | 0,7 | 0,7618 | 0,6754 |
| 17 | 0,2 | 0,8 | 0,7618 | 0,6812 |
| 18 | 0,1 | 0,9 | 0,7804 | 0,6812 |

**Fuente:** (Carvajal, 2012).

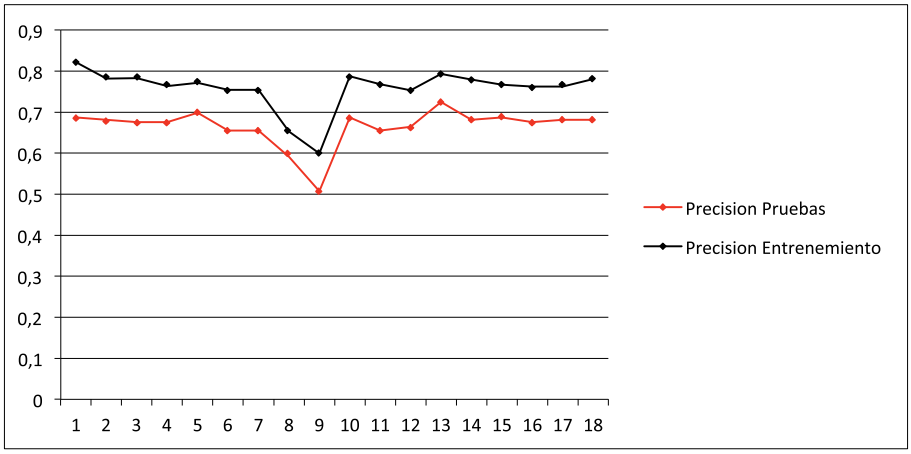


**Figura 3.** Gráfica de las curvas de precisión de pruebas vs. entrenamiento durante todas las ejecuciones.

**Fuente:** (Carvajal, 2012).

Así que el mejor resultado se obtuvo con una población de 600 individuos, 200 generaciones, los valores de los pesos ajustados a 0,5 cada uno, el cruce en 0,6, la mutación en 0,4 y la reproducción (copiado) en 0,1; en este caso la precisión obtenida en entrenamiento fue de 0,7928 y de 0,7246 en pruebas, el tiempo de ejecución del algoritmo fue 8166,741 segundos, el índice de Coheen Kappa fue 0,6829 y el AUC, 0,9589. El modelo obtenido se compone de 29 reglas de clasificación en forma de sentencias IF-THEN que pueden representarse como árboles de decisión, como se muestra en las figuras 4, 5, 6 y 7.


**Figura 4.** Regla de clasificación para el método de ataque PHISHING.

**Fuente:** (Carvajal, 2012).


**Figura 5.** Regla de clasificación para el método de ataque FUERZA BRUTA.

**Fuente:** (Carvajal, 2012).


**Figura 6.** Regla de clasificación para el método de ataque SQL INJECTION.

**Fuente:** (Carvajal, 2012).


**Figura 7.** Regla de clasificación para el método de ataque CROSS SITE SCRIPTING.

**Fuente:** (Carvajal, 2012).

En la tabla 5 se listan las clases y los valores de precisión en clasificación obtenidos por las reglas extraídas para entrenamiento y pruebas.

## CONCLUSIONES

Las reglas obtenidas fueron evaluadas y representadas en árboles de decisión; esto permitió interpretarlas y concluir ciertos comportamientos de interés. Por ejemplo, la figura 7 indica que para CROSS SITE SCRIPTING la herramienta usada no fue INTERCAMBIO DE INFORMACIÓN y la debilidad de la aplicación fue un MANEJO IMPROPIO DE SALIDAS. Además se filtraron patrones de comportamiento obvios como la intencionalidad de los hackers y terroristas o el objetivo común de los ataques de phishing. A continuación se describen los patrones encontrados.

- Las entidades más susceptibles a ataques de secuestro de sesión pertenecen al gobierno o al ejército y las que menos son atacadas son las organizaciones sin ánimo de lucro; estos ataques buscan una ganancia financiera y son perpetrados generalmente por espías.
- Las entidades menos afectadas por los virus son negocios que no se dedican al comercio y/o actividades financieras, como empresas de tecnología, medios de comunicación o cadenas de hoteles.
- Las entidades más vulnerables a los ataques de denegación de servicio son las instituciones de salud. Además la herramienta predilecta para estos ataques son los toolkit.
- Las entidades menos afectadas por los troyanos son las entidades de comercio o venta al por menor.
- Los ataques de *Cross Site Scripting* no son responsabilidad de usuarios inexpertos, es decir, no corresponden a errores humanos, ya que la debilidad de la aplicación es *Manejo Impropio de Salidas*, lo cual indica un error en la construcción del software o en el diseño del mismo.
- Los ataques de *SQL Injection* son ocasionados por un manejo impropio de entradas, recayendo la responsabilidad en los desarrolladores del software al no seguir estándares de construcción que protejan los campos de entrada de las aplicaciones.
- Los incidentes de *Localización Predecible de Recursos* ocurren por falta de restricciones al usuario y las entidades menos susceptibles a este tipo de ataques pertenecen al gobierno o al ejército.
- Los atacantes en los incidentes de *Acceso no Autorizado* son en su mayoría terroristas usando comandos de usuario.
- El *abuso de funcionalidad* es perpetrado por hackers en la mayoría de los casos.

La fiabilidad de estos patrones depende del porcentaje de precisión alcanzado por las reglas de clasificación; las clases que obtuvieron mayor precisión fueron: SQL Injection, Divulgación no Intencional, Cross Site Scripting, Denegación de Servicio y Troyano. En general, aquellas clases que presentaban más de 120 instancias en el repositorio obtuvieron reglas con una precisión superior a 79%. Para las clases con un número menor de instancias se obtuvieron reglas con precisiones entre 77% y 22%, a excepción de las clases Virus y Abuso de Funcionalidad, que tenían 34 y 24 instancias, respectivamente, y cuyas reglas tuvieron 0% de precisión; esto se contrasta con clases con igual número de instancias como Localización Predecible de Recursos, Spyware o Phishing, que lograron 50%, 65% y 61%.

Solo 4 clases tenían más de 120 instancias en el repositorio, el promedio del número de registros por clase fue 41,13, mientras que la desviación estándar para el mismo concepto fue 68,03, así que la mayoría de clases tenía un número mucho menor que 120 registros; no obstante, el modelo de clasificación alcanzó una precisión aceptable tanto en clasificación como en pruebas (79,28% y 72,46%, respectivamente).

De lo anterior se concluye que la programación genética y más específicamente el enfoque de K.C. Tan logra excelentes resultados cuantas más instancias por clase se le proporcionen al algoritmo para el aprendizaje, pero no depende directamente de esta cantidad de registros, pues el algoritmo logra descubrir reglas con calidad aceptable (superior a 75%) para clases con un número precario de instancias.

Por otra parte, es fundamental la obtención de las reglas en sentencias de tipo IF-THEN, ya que su interpretación hizo posible encontrar los patrones de comportamiento de los datos que conforman el repositorio; esta interpretación no es tan viable en otro tipo de modelos como el entregado por redes neuronales o vecinos próximos, una ventaja que presenta este modelo de obtención de reglas.

Los tiempos de ejecución del algoritmo fueron altos en su mayoría (más de 90 minutos); vale la pena ahondar en el desarrollo de herramientas para PG, realizando implementaciones en lenguajes más simples que JAVA, aún más cuando la labor de minería de datos muestra preferencia por lenguajes como Python o R (Harvard, 2013).

Sigue siendo un obstáculo encontrar los mejores valores para los parámetros del algoritmo, donde ya se han hecho avances (Yuan, 2005; Yang, 2000), pero que en algunos casos como el del problema tratado llegan a ser muy costosos en tiempos y recursos, como lo sería diseñar un ANOVA.

Finalmente, es importante reconocer la madurez que ha alcanzado la programación genética, la cual mostró resultados satisfactorios en cuanto a la obtención de patrones para modelar el comportamiento de los incidentes de seguridad, siendo este repositorio un conjunto de datos no numéricos que no permite otro tipo de análisis como la estadística típicamente descriptiva. Este método de obtención de reglas mediante el enfoque de K.C.Tan puede usarse en otros problemas en donde la calidad de los resultados terminará dependiendo del correcto ajuste de los parámetros y del número de registros por clase que se le proporcione al algoritmo para su aprendizaje.

## FINANCIAMIENTO

Este artículo es fruto de la investigación de proyecto de grado y no tuvo ningún financiamiento externo además del de los autores.

## REFERENCIAS

Banzhaf ,W.; Nordin, P.; Keller, R. E. y Francone , F. D. (1998). *Genetic Programming: An Introduction – On the Automatic Evolution of Computer Programs and Its Applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers.

Bojarczuk, C.C.; Lopes, H.S. & Freitas, A.A. (2004). A constrained-syntax genetic programming system for discovering classification rules: application to

medical data sets. *Journal Artificial Intelligence in Medicine*, Vol. 30, ene., pp. 27-48.

Carvajal, C.J. y Bayona, D.N. (2012). Análisis de Incidentes de Seguridad Informática Mediante Minería de Datos, para Modelado de Comportamiento y Reconocimiento de Patrones. Tesis de ingeniería de sistemas no publicada, Universidad Francisco José de Caldas, Bogotá, Colombia.

De Falco, I.; Della Cioppa A., y Tarantino, E. (2001). Discovering interesting classification rules with genetic programming. *Applied Soft Computing*., Vol. 1, No. 4, May, pp. 257-269.

Freitas A.A. (2002). *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Han, J. & Kamber, M. (2005). *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers.

Harvard School of Engineering and Applied Sciences (2013). CS109 Data Science. Recuperado de: http://cs109.github.io/2014/

Knowledge Discovery and Intelligent Systems KDIS (2011). Java Class Library for Evolutionary Computation JCLEC. Recuperado de: http://jclec.sourceforge.net/

Koza, J.R. (1992). *Genetic Programming: On Programming Computers by means of Natural Selection and Genetics*. Cambridge, MA, USA: MIT Press.

Luna, J.M.; Romero, J.R. y Ventura, S. (2012). Design and Behaviour Study of a Grammar Guided Genetic Programming Algorithm for Mining Association Rules.

*Knowledge and Information Systems*, Vol. 32, Jul., pp. 53-76.

Mendes, R.R.; Voznika, F. de B.; Freitas, A.A. & Nievola, J.C. (2001). Discovering Fuzzy Classification Rules with Genetic Programming and Co-evolution. *PKDD '01 Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery,* pp. 314-325.

Poli, R.; Langdon, W.B. & Mc Phee, N.F. (2008). *A Field Guide to Genetic Programming*. UK: Lulu Enterprises.

Privacy Rights Clearing House (2010). Chronology of Data Breaches 2005-Present. Recuperado de: http://www.privacyrights.org/data-breach

Software Engineering Institute Carnagie Mellon (2010). CERT Statistics (Historical). Recuperado de: http://www.cert.org/stats/cert_stats.html

Tan, K.C.; Tay, A.; Lee T.H. & Heng, C.M. (2002). Mining Multiple Comprehensible Classification Rules Using Genetic Programming. *Proceeding CEC '02 Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress*, Vol. 2, pp. 1302-1307.

The Web Application Security Consortium (2010). Web Hacking Incident DataBase. Recuperado de: http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database

Universidad Distrital Francisco José de Caldas (2010). COL-CSIRT Grupo de Investigación. Recuperado de: http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/colcsirt/

Wong, M.L. & Leung , K.S. (2000). *Data Mining Using Grammar Based Programming and Application*. Norwell, MA, USA: Kluwer Academic Publishers.

Yang, Q.W.; Jiang, J.P. & Chen, G. (2000). How to Select Optimal Control Parameters for Genetic Algorithms. *Proceedings of the 2000 IEEE International Symposium on Industrial Electronics*, Vol. 1, Dec., pp. 37-41.

Yuan, B. & Gallagher, M. (2005). A Hybrid Approach to Parameter Tuning in Genetics Algorithms. *The 2005 Evolutionary Computation IEEE Congress*, Vol. 2, Sep., pp. 1096-1103.