# Técnicas de Datamining aplicadas al procesamiento de Logs

Luciano M. Guasco Javier Echaiz Jorge R. Ardenghi

Laboratorio de Investigación de Sistemas Distribuidos (LISiDi)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur, Bahía Blanca (8000), Argentina
{lmg,je,jra}@cs.uns.edu.ar

**Resumen**

En esta línea de investigación se establecen algunas técnicas que presentan soluciones eficientes a problemas presentados en [1].

El problema principal es el procesamiento de los *logs* de un nodo determinado. Los *logs* se presentan en forma de archivos de texto, y documentan eventos correspondientes a los servicios que se ejecutan en un host. Esta información existe de forma masiva, desordenada y poco legible para un administrador de sistemas y guarda información relevante sobre posibles ataques.

Se busca encontar un método para procesar esta información y convertirla en reglas de conocimiento, útiles en el framework de [1]. Con técnicas de datamining, es posible hacer un proceso efectivo y de alta performance de esta información. Se presentan entonces varios dominios de datos a explorar en un host y las características a ser clasificadas por las técnicas mencionadas.

## 1. Introducción

En la línea de investigación presentada en [1], se ha establecido un framework para la Detección de Intrusos basado en la representación de conocimiento por parte de los nodos integrantes de un dominio. Este conocimiento es utilizado para crear un debate entre los nodos acerca de un posible ataque, utilizando argumentación, a través de la programación en lógica rebatible (Defeasible Logic Programing, DeLP) [2].

El tema que se presenta en este trabajo, es consecuencia de un problema a resolver para llevar a cabo la propuesta anterior. Uno de los principales inconvenientes es poder seleccionar información relevante, no redundante, y que aporte datos precisos al mecanismo de inferencia presentado en [1].

En [1], se había propuesto utilizar algunos scripts, para procesar directamente los *logs* de los servicios, y esta información transformarla en reglas que eran insertadas en la base de conocimiento común, y desde la cual se hacía la inferencia. Sin embargo, trabajar con los datos sin procesar hacía que las reglas sean muchas y no estén optimizadas, lo cual presentaba problemas de performance.

Para solucionar esto se presentan varios métodos, utilizando técnicas de datamining [3], para que cada host pueda recolectar, desde la información *raw* que este maneja, un resultado más preciso para luego aportarlo a la base de conocimiento y al mecanismo de inferencia de [1].

La elección de utilizar técnicas de datamining, recae en que los *logs* se presentan en forma de datos masivos, sin un criterio de selección previo, y de los cuales cuesta mucho obtener información útil. Además, los modelos probabilísticos que se crean reportan un alto grado de exactitud en la clasificación de los datos si se seleccionan clasificadores adecuados para el dominio de datos.

## 2. Diferentes modelos para el procesamiento de logs

Como se explica anteriormente, existe la necesidad de que un host pueda determinar un posible ataque, para luego argumentar esta acción con el resto de los integrantes del dominio y allí determinar la veracidad de este hecho.

Para esto se crearán distintos modelos de clasificación, que luego de una etapa de aprendizaje de las conductas normales, puedan determinar posibles ataques a partir de una clasificación de los *logs*

del sistema, que representan las conductas que un cliente o usuario está llevando a cabo con el nodo particular.

La idea principal, es tener varios módulos de aprendizaje y clasificación, que puedan integrarse para formar un método sólido para un host, en la detección de posibles atacantes. Estos módulos representan distintos dominios de datos sobre los cuales se trabajará.

La abstracción en módulos permite hacerlos configurables para las necesidades particulares de los administradores de sistemas, sin tener que modificar directamente los clasificadores. La intención es que a través de la especificación del dominio de datos, se pueda entrenar de forma dinámica a los clasificadores, para tener un modelo de clasificación perfectamente adaptado según las necesidades de cada nodo.

A continuación se presentarán distintos dominios de datos que pueden inspeccionarse con técnicas de datamining, para obtener resultados útiles que serán presentados a la base de conocimiento de [1].

### 2.1. Secuencia de acciones

Este módulo brindará la funcionalidad de tener un clasificador que, una vez entrenado, pueda determinar si una cierta secuencia de acciones (interacciones cliente/servidor) a un servicio determinado, que realiza un cliente con el nodo, representa un ataque.

Con acciones nos referimos a interacción entre el usuario y algún servicio disponible en el nodo. Todas las acciones se llevan a cabo a través de la red que conecta cada nodo con el resto y con la red exterior (Internet, otra LAN, etc.).

Por ejemplo, dados:

- $t_1$ $accion_A$: Conexión HTTP al puerto 80.
- $t_2$ $accion_B$: Escaneo de los puertos abiertos del nodo (herramientas como por ej. nmap).
- $t_3$ $accion_C$: Intento de conexión fallido SSH al puerto 22.
- $t_4$ $accion_D$: Intento de conexión fallido SSH al puerto 22.
- $t_5$ $accion_E$: Intento de conexión fallido FTP al puerto 21.

Donde cada $accion_X$, representa una acción en un tiempo $t_i$ determinado que realiza un cliente o usuario interactuando con un nodo (en este caso un servidor).

Sería factible que esta secuencia de acciones en un lapso de tiempo determinado, se clasifique como un posible ataque, dado que la mayoría de los atacantes verifican el dominio con una conexión *web* (HTTP) y luego prueban distintas vulnerabilidades.

Este planteo implica que entrenemos el módulo para crear el modelo del clasificador. Para esto, es necesario tener conexiones seguras, que determinen las secuencias de acciones que usualmente realizan los clientes con el nodo.

### 2.2. Ratio *input / output*

Un buen método para la detección de posibles ataques es utilizar como dominio de datos la proporción de *input / output* en un servicio determinado. La idea principal es detectar cuando se pasa algún umbral del *ratio input / output* . Este umbral será determinado dinámicamente con el transcurso del tiempo, y cada vez que se vaya entrenando el clasificador que lo componga.

La efectividad de este módulo recae en que existe un patrón determinable en la proporción de datos de entrada sobre datos de salida, para un determinado servicio según la funcionalidad del mismo.

Por ejemplo, supongamos que tenemos un servidor Web, HTTP 80, y un sitio alojado en el servidor: www.vidscience.com.ar, y este sitio contiene una colección de *streams* de video. Entonces podemos crear un clasificador, que luego de entrenarlo con conexiones seguras al sitio, puede determinar los

umbrales normales de *ratio input / output* de conexiones con el sitio. Supongamos también que un atacante quiere correr un *exploit* sobre la videoteca mencionada, un simple script que se ejecute en el *server* remotamente con fines maliciosos. El hecho de ejecutar un *exploit* en el servidor *web*, sobre la página de videos, implica que exista una conexión desde el cliente al servidor, con un *input* de datos considerable (un script a ejecutar), y un *output* despreciable (una vez ejecutado el script, no se espera resultado por parte del *server*). Luego, tenemos una conexión con un *ratio input / output* sobre la videoteca, que no es probabilísticamente normal, según lo aprendido por el clasificador, ya que este *ratio* de conexiones con la página será de un *input* bajo (los accesos a los videos) / *output* alto (el stream de datos de cada video solicitado enviado al browser del cliente) con una probabilidad muy alta.

### 2.3. Conexiones globales

Otro módulo a considerar, es un clasificador que determine que el *ratio* de conexiones, y de datos *input / output* generales en el servidor ha pasado el umbral de lo normal. Es una generalización de lo mencionado en el inciso anterior.

Esto es útil cuando tenemos un servidor que corre muchos servicios, y tiene múltiples accesos desde el exterior. Si se entrena un clasificador durante mucho tiempo con conexiones seguras y normales del servidor, se puede ver que existe un patrón en los *ratios* de conexiones *input / output* , según la hora, o la época del año. Así, podemos entrenarlo para determinar si estos valores luego de ser clasificados, pueden deberse a un posible ataque o vulnerabilidad en el sistema. Nótese que este tipo de clasificador no determina específicamente un atacante particular, pero si es una alarma válida cuando el servidor está bajo ataque.

### 2.4. Secuencia de accesos

Una característica importante de los ataques a los sitios *web*, es la inspección previa del sitio en busca de vulnerabilidades. En el caso del ataque mediante *SQL injection*, como se menciona en [4], el cliente debe recorrer la página en busca de un acceso de *login*, o alguna consulta posible al servidor. Es interesante entonces inspeccionar los patrones de acceso a las páginas que tienen estos atacantes.

Por ejemplo, supongamos que tenemos un sitio *web* de venta de productos electrodomésticos, www.miselectrodomesticos.com, y que tiene un *link* desde la página principal al sitio del administrador. Además la página principal brinda un *link* para el personal que se encarga de mantener el *stock*, y los precios *online*. Ya que los usuarios tienen promociones y descuentos, según el historial de compras en el sitio, entonces también tienen cuentas en la misma. Estos accesos de los compradores, el personal de *stock*, y el administrador, requieren un *login* mediante *user* y *password*. Entonces un posible atacante, paseando por nuestro sitio, y tratando de ejecutar una inyección SQL, en los sitemas de logueo, podría realizar la siguiente secuencia de *links*:

1. www.miselectrodomesticos.com
2. Login Failed: www.miselectrodomesticos.com/Administrator
3. Login Failed: www.miselectrodomesticos.com/Usuarios
4. Login Failed: www.miselectrodomesticos.com/Stock

Podemos entonces desarrollar un clasificador, que a partir de estas secuencias de accesos, o *links*, determine un posible ataque, o una navegación normal del sitio. Para esto, podemos utilizar los *logs* de acceso a las páginas del servidor *web*.

### 2.5. Secuencia de Syscalls

Otro ataque clásico es el *command masquereade*, o enmascarado de comandos [5]. La idea es que un comando clásico del sistema operativo, enmascare código malicioso. Para esto, se debe tener acceso previo para poder reemplazar el ejecutable nativo por el nuevo binario infectado. Algunos shells presentan alguna vulnerabilidades a estos tipos de ataques, y permiten ejecución desde los directorios temporales, que mediante *links* simbólicos desde el path correspondiente ejecutan el código allí ubicado.

Como no existe forma de decidir si un comando no está haciendo nada más que lo esperado, podemos realizar un módulo que detecte con una probabilidad alta estos tipos de ataques.

Muchos estudios se han realizado en este tipo de ataques, utilizando técnicas de *Machine Learning*, como clasificadores Bayesianos, por ejemplo en [5].

Una extensión del uso de clasificadores Bayesianos para la detección de ataques de enmascarado de comandos puede hacerse considerando los *systemcalls* que cada comando realiza. Cada comando ejecuta, según los parámetros del mismo y las opciones correspondientes, una secuencia de *systemcalls* que son directivas al sistema operativo para poder llevar a cabo la ejecución de los comandos del shell. Es posible desarrollar un análisis sobre un posible ataque si se considera la secuencia de estos *systemcalls* que se ejecutarán para un determinado comando a ejecutar a partir del *shell*.

## 3. Características de los módulos

La decisión a tomar antes de comenzar el trabajo será sobre los clasificadores. Es necesario un análisis profundo de los distintos tipos de clasificadores, y de los dominios de datos que se están utilizando, para evaluar los mejores resultados.

Existen varios tipos de clasificadores pero los Bayesianos, los árboles de decisión, y las reglas, son los que mejor se adaptarán, según las especificaciones antes mencionadas. Ya varios trabajos se han realizado con estos clasificadores, que pueden servir para especular óptimos resultados y hacer una elección correcta, como en [6, 7, 8].

Cualquiera sea el tipo de clasificador seleccionado, lo que hay que tener en mente es que estos clasificadores se utilizarán para formar un modelo dentro de un módulo, que tendrá como datos de entrada distintos *logs* de servicios, que podrán ser transformados si el clasificador lo requiere (normalizados, discriminados, etc.), y como salida generarán reglas que formarán parte de la base de conocimiento presentada en [1],para luego ser utilizadas en una argumentación acerca de un posible ataque.

El tipo de módulos antes mencionado contará con la facilidad de adaptación y configuración para distintos ambientes de trabajo. Los módulos se presentaron en este trabajo con ejemplos simples, en general sobre servicios *webs*, pero cabe notar que esta idea es aplicable a cualquier dispositivo o nodo en la red del dominio. En [1], el framework analizaba distintas posturas frente a posibles ataques por parte de múltiples nodos y dispositivos. Esto requiere que los módulos sean adaptables a *routers*, *firewalls*, etc.

## 4. Escalabilidad y Performance

Este trabajo presenta un manejo eficiente de los logs, dejando atrás alternativas presentadas en [1], donde los datos eran presentados a la base de conocimiento sin ser procesados, y con la necesidad de una optimización y un podado previo de las reglas.

En el marco del trabajo [1], se ve la posibilidad de llevar el framework a un ambiente grid. Es necesario entonces mantener la performance y la robustez de la aplicación.

Existe la necesidad de realizar un análisis de los distintos clasificadores que se usarán en los módulos, para seleccionar el que obtenga mejor performance y exactitud. Para esto pueden usarse técnicas conocidas de evaluación y comparación de clasificadores para un dominio de datos determinado, como se postula en [9].

## 5. Trabajos Futuros

Este trabajo explora un área poco profundizada y muy importante por su repercusión en la seguridad en sistemas, agregando independencia en la administración de sistemas. Al momento de llevar este trabajo a formar de la implementación de [1], deberáhacerse un análisis más profundo de los temas aquí mencionados tratando incluso de generar nuevos clasificadores, que lleven a la efectividad de la clasificación al máximo para estos dominios de aplicación.

El siguiente paso en esta línea de investigación recae en la implementación del sistema mencionado en [1], pero utilizando técnicas como las mencionadas en este trabajo, para mejorar la perfomance en el procesamiento de *logs*. Además se abarcarán todos los problemas nuevos que surjan en esta adaptación, siempre buscando mejorar la performance y la eficiencia en la detección de intrusos.

Otro aspecto interesante a realizar es la adaptación de estos módulos para que todo el modelo pueda ser utilizado en servidores distribuidos, que cada día se vuelven más populares por su poder de cómputo. Notemos que el esquema distribuido requiere un análisis de los*logs*de forma remota, y que se incrementan los puntos de falla del sistema. Esto nos lleva a analizar factores como la tolerancia a fallas y la necesidad de replicación de datos.

En áreas de aplicaciones para grids con la utilización de herramientas de datamining, como en [10], y áreas de sistemas distribuidos aplicados al cómputo de herramientas de datamining, como en [11], la seguridad no se vio como un aspecto importante en los últimos años, y ahora comienza a ser un tema de alta prioridad. Por eso, este trabajo abre las puertas a inspeccionar muchos aspectos de *Grid Computing, Datamining* y Seguridad en Sistemas, y deja como tarea futura, el análisis detallado de las características presentadas, para poder llevar a cabo la mejor implementación de [1].

## Referencias


[1] Luciano M. Guasco, Javier Echaiz y Jorge Ardenghi, "Framework para Detección de Intrusos utilizando DeLP," *WICC*, 2007.

[2] A. J. García and G. R. Simari, "Defeasible Logic Programming: An Argumentative Approach," *TPLP*, vol. 4, no. 1-2, pp. 95–138, 2004.

[3] T. M. Mitchell, "Machine learning and data mining," *Communications of the ACM*, vol. 42, no. 11, pp. 30–36, 1999.

[4] William G.J. Halfond, Jeremy Viegas, and Alessandro Orso, "A Classification of SQL Injection Attacks and Countermeasures," *International Symposium on Secure Software Engineering*, 2006.

[5] Roy A. Maxion and Tahlia N. Townsend, "Masquereade Detection Using Truncated Command Lines," *International Conference on Dependable Systems and Networks*, 2002.

[6] Junbing He, Dongyang Long, Chuan Chen, "An Improved Ant-based Classifier for Intrusion Detection," *Third International Conference on Natural Computation (ICNC)*, 2007.

[7] J. Gomez, D. Dasgupta, "Evolving Fuzzy Classifiers for Intrusion Detection," in *Proceedings of the IEEE*, 2002.

[8] Paul Dokas, Levent Ertoz, Vipin Kumar, Aleksandar Lazarevic, Jaideep Srivastava, Pang-Ning Tan, "Data Mining for Network Intrusion Detection," 2002.

[9] T. Mitchell, *Machine Learning.* McGraw Hill, 1997.

[10] "www.datamininggrid.org," *Data Mining Tools and Services for Grid Computing Environments.*

[11] I. Janciak, M. Sarnovsky, A. M. Tjoa, and P. Brezany, "Distributed classification of textual documents on the grid," in *HPCC*, pp. 710–718, 2006.