

Análisis de datos procedentes de un Sistema de Detección de Gusanos mediante técnicas de clustering

Urko Zurutuza¹, Roberto Uribeetxeberria, y

Miguel Fernández

Dept. de Informática
Mondragon Goi Eskola Politeknikoa
Mondragon Unibertsitatea
20500 Mondragon

{uzurutuza, ruribeetxeberria, mfernandez}@eps.mondragon.edu

Diego Zamboni

IBM Research GmbH
Zurich Research Laboratory
CH-8803 Rüschlikon
Switzerland
dza@zurich.ibm.com

Resumen

En este trabajo se muestra un marco que puede ser utilizado para analizar datos provenientes de Sistemas de Detección de Gusanos (WDS ó *Worm Detection System*) como *honeypots*, o *network telescopes* mediante técnicas de *data mining*. La aplicación de estas técnicas ayuda a la comprensión de la naturaleza de un gran volumen de tráfico malicioso que circula por Internet, además de detectar fallos en la configuración de dispositivos. A lo largo del trabajo se muestra cómo se exploran, diseñan e implementan técnicas de *data mining* para analizar los datos producidos por un WDS.

1. Introducción

En la actualidad, los daños generados por un único atacante de forma manual no son tan devastadores como los que provocan los ataques automáticos a gran escala, más conocidos como virus y gusanos. En [7] tratan de modelar el daño que podría producir un atacante con el objetivo de infectar el mayor número de sistemas posibles en EEUU causando el máximo perjuicio posible en cada sistema. Según sus estimaciones, en el peor de los casos, el gusano (con un contenido altamente destructivo) podría causar un daño económico superior a los 50 billones de dólares atacando sólo los sistemas Windows. Estos datos no hacen más que confirmar el peligro que acecha

a los sistemas informáticos de hoy en día, y más teniendo en cuenta que la motivación de los atacantes también ha cambiado, desde la realización de ataques para obtener renombre y admiración, hacia el uso de *botnets* por parte de mafias y grupos organizados con ánimo de lucro controlando redes de ordenadores *zombies* con el propósito de lanzar *spam*, realizar ataques de denegación de servicio y otras muchas acciones ilegales.

Por todo ello se hace necesario avanzar en la investigación en torno a detección de intrusiones y buscar nuevos métodos que permitan detectar, analizar, y modelar los ataques lo antes posible con el fin de desarrollar métodos de alerta temprana y también de protección ante los ataques automáticos a gran escala.

2. Sistemas de detección de gusanos

Los WDS se pueden considerar Sistemas de Detección de Intrusiones (IDS) especializados en detectar este tipo de ataques, por lo que las técnicas utilizadas para su detección, así como la clasificación de estos sistemas no difieren mucho de los IDS tradicionales, aunque sí tienen ciertas particularidades asociadas a la naturaleza de dichos ataques.

En 1988 se descubrió el primer gusano que se propagaba por Internet. Este gusano se conoce como el Gusano Internet o el Gusano Morris. En 1988 el profesor Spafford escribió un informe analizando dicho programa [2]. En el informe, define gusano como un programa que puede funcionar de forma autónoma y propagar una versión completamente funcional de sí mismo a otras máquinas. Se podría clasificar a los WDS en

¹ Este autor está financiado por el Gobierno Vasco, a través de la beca BFI 05.454 del Programa de Formación de Investigadores del Departamento de Educación, Universidades e Investigación.

función de su fuente de información, método utilizado para el análisis, y respuesta que dan ante un gusano detectado.

2.1. Fuentes de información y monitorización

Las fuentes de información que se manejan para la detección de gusanos siguen vigentes; por un lado datos recogidos de una *red* (tráfico completo de una red, *ICMP destination unreachable* proveniente de enrutadores) para luego analizar dichos datos en busca de gusanos, y por el otro lado datos recogidos a nivel de *host*, como registros de auditoría, llamadas al sistema o datos de ejecución en memoria.

Como sistemas de monitorización, en los últimos años han surgido dos sistemas muy apropiados para monitorizar gusanos: *honeypots* y *network telescopes*. Así, ya desde 1992 se planteó el uso de los *honeypots* o sistemas trampa para el engaño, seguimiento y aprendizaje de atacantes [4]. Un *honeypot* es un señuelo en forma de recurso informático vulnerable utilizado para distraer atacantes, servir como un recurso de alerta temprana sobre nuevas técnicas de ataque, y facilitar el análisis en profundidad sobre los atacantes [5].

El uso de estos sistemas, enfocados a monitorizar actividades derivadas de ataques automáticos basados en el escaneo aleatorio o semi-aleatorio, tiene la particularidad de asignar a los señuelos espacios de direcciones IP no asignadas de modo que todo intento de conexión a dichos *honeypots* será automáticamente considerado como sospechoso. Por otro lado, el nivel de interacción de un *honeypot* es un aspecto fundamental. Cuanta mayor interacción ofrezca el sistema (e.g.: respuestas a conexiones TCP), mayor cantidad de datos se podrá capturar y por lo tanto mayor conocimiento se tendrá del ataque.

2.2. Análisis

No se debe olvidar que los IDS tradicionales son aptos para la detección de gusanos, ya que en el caso de sistemas de detección de uso indebido se incluyen firmas para la detección de gusanos (productos antivirus, *Snort* [6], *Bro* [11]), y los sistemas basados en anomalías son perfectamente capaces de detectar las anomalías que generan los gusanos en el tráfico de red.

Los gusanos que operan escaneando de forma aleatoria resultan más fáciles de detectar debido a que se conoce de antemano su modo de funcionamiento. Así, mediante el uso de monitores en direcciones de red no asignadas se pueden detectar de un modo sencillo y preciso.

Aún y todo, y al igual que ocurre con los IDS tradicionales, son los basados en firmas los que más se utilizan en el mercado actual. Haciendo uso de técnicas de reconocimiento de patrones son capaces de detectar gusanos conocidos. Si bien es verdad que el desarrollo de las firmas de ataques consumen tiempo y recursos, se producen firmas lo suficientemente precisas como para detectar gusanos con muy poca tasa de falsos positivos. Una técnica que alienta a seguir utilizando este tipo de sistemas es la generación automática de firmas, técnica que cada vez está tomando mayor relevancia en el área de investigación.

2.3. Respuesta

Son varias las técnicas utilizadas para tratar de contener lo antes posible la propagación de un nuevo gusano. En [8] se da una buena perspectiva de técnicas utilizadas, clasificando las defensas en soluciones basadas en limitar recursos (cuarentena, regular o limitar el tráfico), soluciones basadas en trabajos coordinados de detección y prevención, soluciones preventivas, y finalmente soluciones de combate móvil (estrategias activas como interceptación, parcheo y anti-gusanos).

Por otro lado, recientemente se han propuesto una serie de trabajos de generación automática de firmas de gusanos a partir de datos recogidos en *honeypots* o directamente de la red para alimentar a los sistemas de detección. Si bien esta técnica no es nueva, ya que se han realizado buenos trabajos en el área de detección de intrusiones, es ahora cuando se está tomando en mayor consideración para la defensa activa de gusanos.

3. Data mining

Desde 1998 con el trabajo de Wenkee Lee y Sal Stolfo [12] hasta ahora, numerosas investigaciones se centran en la aplicación del *data mining* en el campo de la detección de intrusiones. Así, se han utilizado gran cantidad de algoritmos para muchas de las funciones

existentes en el *data mining*, como la clasificación (aprendizaje supervisado) y *clustering* (no supervisado), aunque también se pueden encontrar otras aplicaciones. En [10] se recogen los trabajos más relevantes en esta área.

El conocido como KDD (*Knowledge Discovery in Databases*), se refiere al proceso no trivial de identificar patrones válidos, nuevos, potencialmente útiles y comprensibles a partir de datos [9]. *Data mining* es la aplicación de algoritmos específicos para la extracción de patrones o modelos de los datos. Es un paso particular del proceso de KDD. Para la realización de cualquier proceso KDD, a menudo se siguen las siguientes tareas:

1. Aprendizaje del dominio de aplicación. Se requiere para decidir qué atributos pueden ser determinantes a la hora de crear los modelos deseados. También incluye el estudio de la tipología de datos con los que se pretende trabajar y el tipo de modelos que se desean obtener. Este paso sirve de guía a la hora de realizar las siguientes tareas del proceso.
2. Crear un conjunto de datos objetivo. Incluye tomar la decisión sobre qué atributos utilizar o qué subconjunto de variables serán utilizadas. También implica la selección de muestras de datos a modo de entrenamiento y testeo.
3. Preparar los datos. Este paso tiene por objetivo obtener un conjunto de datos consistente, limpiando y transformando los datos. Implica eliminar ruido (si así se desea), decidir estrategias a la hora de tratar valores nulos, transformación de tipos de datos o incluso de atributos mediante categorías con el fin de preparar los atributos para el paso siguiente.
4. *Data mining* se refiere a decidir el propósito del modelo que se extraerá (como clasificación, agrupamiento, reglas de asociación,...), escoger el algoritmo de *data mining*, establecer los parámetros adecuados del algoritmo, y la propia ejecución del mismo para obtener los modelos.
5. Interpretación y evaluación de los modelos o patrones. Los modelos obtenidos deben ser analizados e interpretados. Posiblemente, será necesario volver a alguna de las tareas anteriores de forma iterativa realizando un proceso de mejora continua hasta obtener los resultados más adecuados.

Este es el proceso que ha sido utilizado durante los experimentos y análisis de datos del sistema de

detección de gusanos, tal y como se muestra en el apartado 6.

4. Marco de trabajo

En la figura 1 se muestra un diagrama con el marco utilizado en este trabajo. El WDS Billy Goat [1], desarrollado por IBM, interactúa con las máquinas infectadas por gusanos que tratan de propagarse por Internet y recoge el tráfico de red asociado a ellas. Por otro lado, *Snort*, IDS basado en reconocimiento de patrones mediante una base de reglas, corre en el mismo sensor que Billy Goat, por lo que también recoge tráfico asociado a dicha interacción.

Por otro lado, se ha utilizado la herramienta de *clustering* CLARATy [3], también desarrollada por IBM, para el análisis de los datos. CLARATy utiliza el algoritmo de *data mining* AOI (*Attribute Oriented Induction*) para iterativamente generalizar las características del tráfico de red según jerarquías pre-configuradas para cada característica, hasta encontrar un criterio de parada. Su funcionamiento se describe en el apartado 5.2.

Si bien estas técnicas podrían ser utilizadas en cualquier otro sistema con características similares, se ha utilizado Billy Goat como marco para realizar el trabajo.

4.1. Billy Goat

Billy Goat es un Sistema de Detección de Gusanos enfocado en detectar en una red máquinas infectadas con gusanos que tratan de explotar vulnerabilidades conocidas. Es un sistema libre de falsos positivos por defecto, ya que todo lo que recoge puede ser considerado como tráfico ilegítimo. También provee información adicional que puede ser utilizada para detectar gusanos nuevos u otras amenazas emergentes en la red. Billy Goat fue diseñado para aprovecharse de la estrategia de propagación utilizada por la mayoría de gusanos conocidos actualmente. Para descubrir nuevas máquinas a las que infectar, la mayoría de los gusanos tratan de conectarse a direcciones IP escogidas de forma aleatoria o escanean rangos completos de direcciones. De este modo encuentran la mayoría de máquinas en una red, pero también tratan de conectarse a un gran número de direcciones no

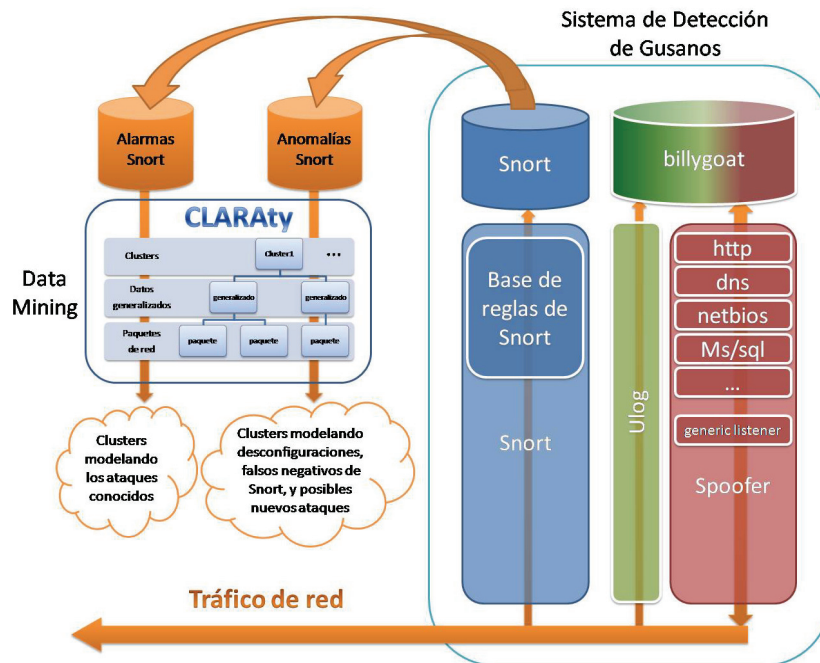


Figura 1. Marco de trabajo.

asignadas. Billy Goat funciona respondiendo a las peticiones de conexión recibidas a direcciones no asignadas fingiendo la existencia de un gran número de máquinas y servicios.

El núcleo de Billy Goat está formado por un mecanismo de virtualización y un repositorio de datos. El primero permite que se puedan escribir servicios individuales utilizando modelos e interfaces estándares, y por consiguiente responder a múltiples direcciones IP de forma transparente. Esto reduce la dificultad de crear nuevos servicios virtuales y su integración con los existentes. La base de datos provee un lugar donde guardar la información asociada a cabeceras IP y detalles de la información a nivel de aplicación generado por los servidores.

Los servicios virtuales ofrecidos por Billy Goat incluyen aquellos que son comúnmente explotados por los gusanos.

4.2. CLARAty

CLARAty es una herramienta utilizada para el *clustering* de alarmas procedentes de IDS desarrollada en IBM. El algoritmo, basado en

Attribute-Oriented Induction (AOI), fue modificado para ajustarse mejor al problema de la agrupación de alertas.

AOI opera sobre una base de datos relacional, y reemplaza los valores de los atributos por unos valores más abstractos repetidamente. Estos valores más abstractos se obtienen de jerarquías de generalización definidas por el usuario. Gracias a la generalización de los valores de los atributos, datos anteriormente diferentes provenientes de Billy Goat se transforman en idénticos y pueden fusionarse creándose así *clusters* o grupos con datos que comparten características similares.

Sin embargo el algoritmo clásico de AOI tendía a sobre-generalizar los atributos, principalmente debido a que en el caso en el que el número de valores distintos de dos atributos era el mismo, generalizaba ambos atributos al mismo tiempo. Por ello, se modificó el algoritmo para adaptarlo a las necesidades. Una de las modificaciones fue la de añadir el parámetro *peso*, el cual influye en el orden en el que los atributos son generalizados durante el proceso de *clustering*. Otra modificación sufrida fue la de añadir un criterio de parada en el proceso de

generalización, añadiendo el parámetro *tamaño mínimo*, el cual mide la fracción de todos los registros necesarios como mínimo para obtener el primer *cluster*. Una vez se ha encontrado un *cluster*, se separa del conjunto de datos y el algoritmo es iterado deshaciendo las generalizaciones previas de los datos restantes. Esta es la tercera modificación realizada en el algoritmo AOI clásico. Como resultado, en el algoritmo 1 se muestra el pseudo-código del algoritmo utilizado en CLARATy como marco de este trabajo.

La herramienta permitió a los autores agrupar alarmas de IDS en *clusters* de forma que todas las alarmas de un *cluster* determinado compartieran la misma causa raíz. Sus experimentos mostraron cómo identificando y removiendo las causas que generaban las alarmas redujeron el volumen de alarmas en un 87%.

Estos resultados tan prometedores obtenidos al utilizar CLARATy para analizar las causas que originan alarmas de IDS, lo hicieron adecuado para su experimentación con datos provenientes de Billy Goat, debido a que ambos casos comparten atributos similares.

```

Input: A table  $T$  and tree-structured generalization hierarchies  $\mathcal{G}_i$ ;
Output: Generalized alarms of user-defined minimum sizes;
Algorithm:
1: for all alarms  $a$  in  $T$  do  $a[\text{count}] := 1$ ; // Initialize counts.
2: Prompt the user for a  $\text{min\_size}$  value;
3: while  $\sum_{a \in T} a[\text{count}] \geq \text{min\_size}$  do {
4:   while identical alarms  $a, a'$  exist do // Merge identical alarms.
5:     Set  $a[\text{count}] := a[\text{count}] + a'[\text{count}]$  and delete  $a'$  from  $T$ ;
6:   if  $\exists a \in T : a[\text{count}] \geq \text{min\_size}$  then { // If we found a solution.
7:     Output the alarm  $a$  and set  $T := T \setminus \{a\}$ ;
8:   }
9:   Replace all  $a' \in T$  by their constituent ungeneralized alarms;
10:  Prompt the user for a new  $\text{min\_size}$  value;
11: } else {
12:   Select an attribute  $A_i$  as specified by Modification 2;
13:   for all alarms  $a$  in  $T$  do // Generalize attribute  $A_i$ .
14:      $a[A_i] := \text{parent of } a[A_i] \text{ in } \mathcal{G}_i$ ;
15: }

```

Algoritmo 1. Algoritmo AOI modificado

5. Resultados

En este apartado se exponen los resultados obtenidos al aplicar CLARATy sobre los datos obtenidos de la interacción entre máquinas infectadas en Internet y el sistema de detección de gusanos. Los resultados se muestran acorde a los pasos típicos de un proceso KDD, tal y como se expone en el apartado 4.

5.1. Aprendizaje del dominio de aplicación

Este paso se ha tratado en el apartado anterior, al mostrar el modo de funcionamiento de Billy Goat, y el modelo de datos utilizado para estos experimentos, que se basa en recoger tanto los paquetes que lanzaron una alerta de Snort, así como todos los paquetes relativos a TCP, UDP e ICMP que contengan datos, es decir, se descartan aquellos paquetes con ventanas relativas a la conexión o desconexión (SYN, FIN, ACK, ...).

5.2. Crear el conjunto de datos objetivo

Para la realización de este experimento, se han definido dos conjuntos de datos diferentes, con objetivos diferentes:

- Conjunto de datos asociado a las alertas producidas por *Snort* (dm_alarms). La aplicación de técnicas de *clustering* sobre estos datos nos permite modelar o aprender los ataques conocidos, y por lo tanto validar la bondad de los *clusters*.
- Conjunto de datos que contiene el tráfico de red cuya IP origen no está involucrada en ninguna alerta de *Snort* ($dm_anomaly$). En este conjunto de datos podemos encontrar datos debidos a fallos en la configuración de dispositivos que envían tráfico de forma descontrolada, y tanto falsos negativos de Snort, como ataques nuevos que tratan de explotar vulnerabilidades conocidas.

Billy Goat recoge un total de aproximadamente un millón de paquetes procedentes de Internet al día. De ese millón, aproximadamente medio son paquetes con datos (descartando así intentos de conexión y escaneos), de los cuales sólo un 5% pertenecen al primer conjunto de datos dm_alarms (paquetes que provocaron una alarma en Snort), y un 20% pertenece al conjunto de datos $dm_anomaly$ (paquetes cuya dirección origen no generaron alarma alguna en Snort). El resto (y la gran mayoría) son datos originados por las máquinas que generaron alguna alarma en Snort. Es decir, podrían ser paquetes legítimos enviados por las máquinas infectadas o atacantes, o también ataques cuyas características no son reconocidas por reglas de Snort.

Para los dos conjuntos de datos escogidos, el vector de características está formado por protocolo, puerto origen, puerto destino, banderas

tcp y datos o *payload*. Si bien las direcciones origen y destino podrían ser variables interesantes a analizar, no se toman en consideración ya que el objetivo es el de conocer la naturaleza del ataque en lugar de conocer cuál es la máquina infectada. Finalmente, la ventana de tiempo escogida es la de un día, debido a que se recogen suficientes datos como para tener una muestra suficientemente representativa.

5.3. Preparar los datos

La preparación de los datos se realiza mediante las jerarquías de generalización utilizadas en cada caso. De este modo, se puede afirmar que el pre-proceso y categorización de los atributos se realiza en tiempo real, mediante la generalización de los mismos. Aún así, es estrictamente necesario pre-definir las jerarquías de generalización de cada atributo de antemano.

Definir las jerarquías de generalización implica codificar el conocimiento previo obtenido en el dominio de aplicación. Obviamente, no hay un modo único de realizarlo. A modo de ejemplo, en la figura 2 se puede observar la jerarquía predefinida para el atributo “puerto destino”.

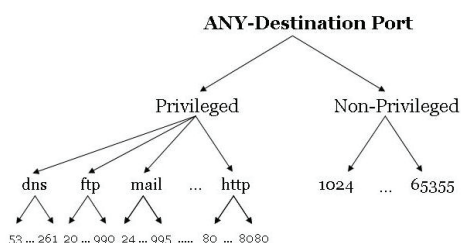


Figura 2. Jerarquía de generalización del atributo “Puerto destino”

5.4. Data mining

En esta tarea se pretende agrupar los datos en función de sus propias características, sin tener un conocimiento previo de a qué tipo de ataque pertenece cada grupo. Por ello CLARATy es una herramienta adecuada cuya validez ya ha sido demostrada con atributos parejos.

Un parámetro muy importante a la hora de configurar la herramienta es la del peso de cada atributo, el cual influencia el orden en el que los atributos son generalizados durante el proceso de

clustering. Conociendo de antemano las características comunes de los gusanos informáticos, se puede concluir que los atributos más críticos, y que por lo tanto más tarde deberían generalizarse son, en este orden: protocolo, puerto destino, puerto origen y banderas. El atributo *payload* no lleva asociado peso alguno, y el cometido del algoritmo de CLARATy es el de obtener las sub-cadenas comunes de cada *cluster* una vez hayan sido agrupadas en función del resto de atributos. De este modo, una vez separados los datos en función de sus características principales, se procede a dividir nuevamente cada *cluster* en función de sus sub-cadenas comunes en el caso de que el *payload* sea diferente.

5.5. Interpretación y evaluación de los modelos obtenidos: alarmas

Para un conjunto de datos de 20089 registros, se obtienen 52 *clusters*. A modo de resumen, en la tabla 1 se muestran los *clusters* más significativos (no se muestra el atributo *payload* por razones de espacio):

Nº	%	Prot	Spt	Dpt	flags
7969	39	udp	non_priv	1434	0
5918	29	udp	non_priv	161	0
1797	8	icmp	8	8	0
1122	5	udp	non_priv	0	0
872	4	icmp	3	3	0
286	1	tcp	non_priv	139	24
222	1	udp	non_priv	137	0
209	1	tcp	non_priv	445	24

Tabla 1. Resumen de los *clusters* obtenidos

Como se puede observar, el cluster más significativo representa el patrón de ataque del gusano *Slammer* también conocido como *Saphire*. En este cluster se encuentran el 100% de los paquetes dirigidos al puerto comúnmente utilizado por *Microsoft SQL Server*. Todo el *cluster* tiene en común el *payload* completo del paquete incluyendo el *exploit* dirigido a ese servicio. El segundo clúster, agrupando un 29% de los datos, representa al servicio *snmp*. Dicho servicio cuenta con multitud de vulnerabilidades no muy graves, pero muy explotadas. También se puede observar que se han obtenidos *clusters* referentes al protocolo *icmp* de tipo *echo* y *destination unreachable* (el tipo de *icmp* se representa mediante el número de puerto origen y destino).

Las alarmas del IDS referidas a estos paquetes prueban que se tratan de posibles *smurf attack* en el primer caso, debido a que dichos paquetes iban dirigidos a la dirección *broadcast*, tratando de provocar que todo el rango responda a la petición. En el segundo caso corresponden a datos conocidos como *backscatter*, o peticiones realizadas a máquinas con la dirección origen alterada de modo que es nuestro sistema quien recibe la respuesta.

Este experimento valida el aprendizaje que se ha realizado sobre el conjunto de datos de alarmas, obteniendo un patrón de los ataques del conjunto de datos objetivo y resumiendo un gran volumen de datos en 52 *clusters*. El marco de trabajo utilizado es válido pues para analizar el siguiente conjunto de datos, y por lo tanto detectar y resumir ataques desconocidos, des-configuraciones y falsos negativos del IDS.

5.6. Interpretación y evaluación de los modelos obtenidos: anomalías

En este segundo conjunto se abordan datos desconocidos, ya que no se conoce la naturaleza de los mismos. El conjunto esta formado por 100575 paquetes provenientes de fuentes sospechosas. Como resultado del *clustering*, se obtienen 96 grupos de datos con características similares. El resumen de los *clusters* más interesantes se puede observar en la tabla 2:

Nº	%	Prot	Spt	Dpt	flags
22002	21	tcp	non_priv	1433	24
19025	18	udp	non_priv	137	0
15522	15	udp	137	137	0
16466	16	tcp	non_priv	139	any
4354	4	tcp	non_priv	445	24
1587	1	icmp	11	11	0
1568	1	icmp	3	3	0
910	0	tcp	non_priv	4662	24
619	0	tcp	non_priv	80	24
603	0	udp	non_priv	53	0
584	0	icmp	8	8	0
357	0	udp	non_priv	38293	0
259	0	tcp	non_priv	4662	25
222	0	tcp	non_priv	4444	24
160	0	tcp	non_priv	1023	any
145	0	udp	non_priv	1434	0
115	0	tcp	non_priv	5554	24
114	0	tcp	non_priv	80	any

Tabla 2. Resumen de los *clusters* obtenidos

Una vez más, el *cluster* más significativo obtenido a partir del conjunto de datos formado por el tráfico asociado a las direcciones origen que no han generado alerta alguna, es el tráfico dirigido al puerto *Microsoft SQL Server*, aunque esta vez se dirige al puerto TCP 1433. Se podría determinar por lo tanto que este tráfico probablemente se refiere al gusano conocido como *Dasher*.

Los dos siguientes *clusters*, referentes al servicio *NetBIOS* de *Windows*, tienen todos los atributos en común salvo el puerto origen. La razón de esta separación en diferentes grupos es que en el conjunto de datos hay suficientes paquetes con el puerto origen 137 como para partir el grupo en dos. Su *payload*, caracterizado con la cadena "CKAA", corresponde con la consulta de recuperación de la tabla de nombres de dicho servicio. Las máquinas *Windows* a menudo intercambian esta información como parte del protocolo de intercambio de ficheros para determinar los nombres *NetBIOS* cuando solo se conoce la dirección IP. Por ello, y aunque este tráfico no suponga un ataque, puede ser utilizado para extraer información útil de las víctimas como nombre de la máquina, dominio, o usuarios conectados. Al ser conscientes de que *Billy Goat* no debería recibir tráfico alguno, todo este tráfico podría tratarse de prolegómenos de un ataque, o bien alguna disfunción de este protocolo perteneciente a máquinas *Windows*.

En los *clusters* obtenidos, también se pueden observar comportamientos interesantes como tráfico dirigido al puerto 4662, utilizada por una conocida aplicación *p2p*, y que por lo tanto podríamos catalogar como configuraciones erróneas. La misma conclusión se podría derivar del tráfico dirigido al puerto 53 o *dns*. En el caso del *cluster* asociado al puerto 38293, utilizado para contactar con servidores *Symantec* para recibir actualizaciones podría también tratarse de este tipo de mala configuración. Finalmente, también se observan *clusters* asociados a actividad de gusanos conocidos como *Blaster* (utilizando el puerto 4444) o *Sasser* (5554).

6. Conclusiones

En este trabajo se ha mostrado un método útil para analizar datos provenientes de *Billy Goat*, si bien

también puede ser utilizado con datos provenientes de otros sistemas similares.

Se ha demostrado que al utilizar CLARATy para analizar los datos de Billy Goat, se detectaron nuevos y comprensivos *clusters*, aunque aun es necesario cierto trabajo manual para analizar los resultados y relacionarlos con comportamientos de gusanos conocidos para identificar y clasificar cada *cluster*.

Por un lado, sirve para resumir en una enorme proporción las alertas de ataques provenientes de IDS tradicionales, pero también para poder analizar y descubrir nuevos tipos de ataques y fallos típicos en la configuración de programas y dispositivos.

7. Líneas futuras

Los resultados obtenidos al agrupar los datos provenientes del conjunto de datos de alarmas han sido comparados con las propias reglas que detectan cada ataque. Los atributos obtenidos de cada *cluster* se asemejan a los atributos utilizados en las reglas, y ello evidencia que el sistema puede ser utilizado en el conjunto de datos con tráfico desconocido para no solo discernir entre ataques o gusanos y fallos de configuración, sino también extraer los atributos de los *clusters* de anomalías en forma de nuevas reglas para la detección de nuevos ataques en IDS tradicionales. Esta es la línea en la que se está trabajando en el momento.

Por otro lado, la plataforma utilizada también monitoriza los intentos de conexión (no sólo a los servicios que simula). Observando las tendencias de intentos de conexión, podremos instalar automáticamente un nuevo servicio genérico en el puerto en el que se detecte la actividad creciente. De este modo, se podría saber si se trata de un escaneo en ese puerto, o si realmente ha emergido una nueva forma de gusano tratando de atacar un servicio vulnerable. Finalmente se podría obtener una regla para dicho ataque emergente.

Referencias

[1] D. Zamboni, J. Riordan y Y. Duponchel. Building and Deploying Billy Goat: a Worm-Detection System. Proceedings of the 18th Annual First Conference. Baltimor (USA), June 2006.

[2] E.H. Spafford. The Internet Worm Program: An Analysis. Purdue Technical Report CSD-TR-823. December 1988.

[3] K. Julisch. Clustering Intrusion Detection Alarms to Support Root Cause Analysis. ACM Transactions on Information and System Security 6 (4), ACM Press, 2003, pp. 443-471.

[4] L. Spitzner. Tracking Hackers. Addison Wesley, ISBN 0321108957, 2002.

[5] L. Spitzner. Honeypots: Sticking To Hackers. Network Magazine, April 2003.

[6] M. Roesch. Snort—Lightweight Intrusion Detection for Networks. Proceedings of USENIX Lisa '99 Conference, Usenix Association, Berkeley, California, 1999.

[7] N. Weaver y V. Paxson. A Worst-Case Worm, Proceedings of the Third Annual Workshop on Economics and Information Security (WEIS04), May 2004.

[8] P. A. Porras, L. Briesemeister, K. Skinner, K. Levitt, J. Rowe, Y.C. Allen Ting. A Hybrid Quarantine Defense. Proceedings of the 2004 ACM Workshop on Rapid Malcode (WORM'04), Washington DC, USA, October 29 - 29, 2004. ACM Press, New York, NY.

[9] U. Fayyad, G. Piatetsky-Shapiro, y P. Smyth. The KDD Process for Extracting Useful Knowledge from Volumes of Data. Communications of the ACM, November 1996/Vol. 39, No. 11.

[10] U. Zurutuza y R. Uribeetxeberria. Revisión del estado actual de la investigación en el uso de data mining para la detección de intrusiones. Actas del I Simposio sobre Seguridad Informática, I Congreso Español de Informática (CEDI 2005). Granada (Spain), Septiembre 2005, pp. 77-84.

[11] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. Proc. 7th USENIX Security Symposium, Jan. 1998.

[12] W. Lee y S. Stolfo. Data mining Approaches for Intrusion Detection. Proceedings of the Seventh USENIX Security Symposium (SECURITY '98), San Antonio, TX, January 1998.