Aplicación de técnicas de agrupamiento a la detección de intrusiones en red mediante N3

Jesús Díaz-Verdejo* Juan M. Estévez-Tapiador Pedro García-Teodoro

Dept.de Teoría de Señal, Telemática y Comunicaciones
Univ. de Granada
ETS Ingeniería Informática
18071 Granada
jedv@ugr.es

Dept. de Informática
Univ. Carlos III de Madrid
Avd. de la Universidad nº. 30
Edificio Sabatini
28911 Leganés (Madrid)
jestevez@inf.uc3m.es

Dept.de Teoría de Señal, Telemática y Comunicaciones
Univ. de Granada
ETS Ingeniería Informática
18071 Granada
pgteodor@ugr.es

Resumen

En el presente trabajo se desarrollan técnicas de agrupamiento de vectores de características para su aplicación en un sistema de detección de intrusiones en red propuesto por los autores. Este sistema, denominado de Vecino Normal más Cercano (N3), proporciona unos excelentes resultados de detección, aunque adolece de un alto coste computacional para su aplicación efectiva en entornos reales. En este trabajo se mostrará que, mediante la aplicación de técnicas de agrupamiento, es posible reducir significativamente la complejidad computacional del sistema, sin degradar los resultados de detección.

1. Introducción

Los sistemas de detección de intrusiones (IDS) basados en red presentan un alto interés como elementos orientados a mejorar la seguridad de las comunicaciones y de los servicios proporcionados por las redes de ordenadores [1]. Frente a los IDS basados en firmas, caracterizados por realizar una comparación de los eventos monitorizados con una base de datos de eventos maliciosos conocidos, los IDS basados en anomalías clasifican los eventos observados como anómalos o normales. Existe, por tanto, un *modelo de normalidad*, a partir del cual se realiza una clasificación que permite detectar los eventos *sospechosos* de corresponder a un ataque.

En este sentido, la potencialidad de los sistemas basados en anomalías para detectar ataques desconocidos y que, por tanto, no podrían ser detectados por uno basado en firmas, se revela como una importante ventaja. Sin embargo, estos sistemas presentan dos dificultades relevantes: la posible clasificación como ataque (o sospechoso) de un evento inocuo, lo que daría lugar a los denominados *falsos positivos* o *falsas alarmas*, y el modelado de la actividad normal del sistema. El primero de los aspectos presenta implicaciones respecto de la aplicabilidad del sistema en entornos reales, ya que una tasa de falsas alarmas moderadamente alta haría que su uso no fuese efectivo para la detección de intrusiones [3]. Por otra parte, el segundo de los aspectos presenta importantes retos investigadores, dado que es necesario aplicar técnicas de modelado que capturen los aspectos relevantes de la operación y permitan la detección de desviaciones respecto del mismo. La mayor dificultad reside en la determinación de los aspectos a modelar, ya que la operación normal de un sistema en red es un fenómeno que presenta una alta complejidad. La mayoría de las aproximaciones existentes abordan el problema de

*Este trabajo ha sido parcialmente subvencionado por el MCYT (proyecto SERVIRA, TIC2002-02798, 70 % fondos FEDER).

Actas del Simposio sobre Seguridad Informática, pp.101-108
ISBN: 84-9732-447-1

forma global, aplicando técnicas de minería de datos o reconocimiento de formas [4].

En el sistema de Vecino Normal más Cercano (N3, de *Nearest Normal Neighbour*) [6] [7], desarrollado por los autores, se propone una aproximación al problema por capas (protocolos), acorde con otras propuestas previas [5], aplicándose técnicas de emparejamiento de patrones para el modelado y la detección. Este sistema proporciona unos buenos resultados de detección manteniendo una baja tasa de falsas alarmas. Sin embargo, su aplicación a sistemas reales resulta dificultada por el coste computacional del algoritmo de detección, que implica una búsqueda sobre los elementos del modelo. En este trabajo se explora la aplicación de algunas técnicas de agrupamiento que permitan reducir el tamaño del modelo y, por tanto, el coste computacional del sistema. Como se explicará más adelante, la aplicación de los algoritmos descritos en la bibliografía no es inmediata, debido a algunas características intrínsecas del sistema N3. Por tanto, se ha desarrollado una adaptación del conocido algoritmo k-medias iterativo que proporciona buenos resultados.

El trabajo se articula en torno al siguiente esquema. En el Apartado 2 se describe brevemente el sistema de detección de intrusiones N3, así como la técnica de modelado utilizada. En el Apartado 3 se discute la aplicabilidad de los algoritmos existentes y se propone una versión de k-medoids iterativo. En el Apartado 4 se describen los resultados obtenidos al aplicar los algoritmos propuestos a N3. Finalmente, se presentan las conclusiones y trabajo futuro.

2. El IDS de vecino normal más cercano

El IDS de vecino normal más cercano, N3 (de *Nearest Normal Neighbour*) [6] opera en base al modelado del tráfico de red a partir de la monitorización de eventos discretos; en particular, de instanciaciones de peticiones correspondientes a un determinado protocolo. Su funcionamiento básico, que se muestra en la Fig. 1, se detalla a continuación. Cada uno de las instancias de tráfico, H, es procesado

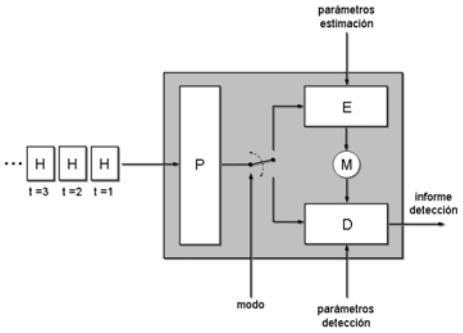


Figura 1: Diagrama de bloques del detector N3.

para obtener la carga útil de las peticiones (bloque P). A continuación, estas cargas útiles son analizadas por el detector (bloque D), que, a partir de la comparación con el modelo de normalidad (bloque M), son clasificadas como normales o anómalas. El modelo de normalidad se obtiene a partir de la monitorización de las cargas útiles recibidas durante un periodo de entrenamiento que debe estar, por definición, libre de ataques.

Los aspectos más relevantes del sistema son la técnica utilizada para el modelado y la comparación con el modelo de normalidad. Esta última se realiza a través del denominado *índice de anormalidad* de la carga útil considerada, p, $A_s(p)$. Para la evaluación de dicho índice se establece una medida de distancia, D, entre dos cargas útiles cualesquiera. Así, el índice de anormalidad de la carga útil p se obtiene como la mínima distancia entre dicha carga útil y un elemento del modelo de normalidad:

$$A_s(p) = \min_{\forall q \in \mathcal{N}_{norm}} \{D(p,q)\} \qquad (1)$$

donde $\mathcal{N}_{norm}$ es el modelo de normalidad, compuesto por cargas útiles normales.

Una vez obtenido el índice de anormalidad, la clasificación de una carga útil se puede realizar sin más que establecer un umbral tal que todas las cargas útiles que superen dicho umbral serán clasificadas como anómalas. De esta forma, la clasificación se realiza, finalmente, en función de la distancia entre la carga útil considerada y la carga útil más próxima en el modelo, lo que implica una búsqueda en el modelo del vecino normal más próximo.

2.1. Distancias entre cargas útiles

La distancia entre dos cargas útiles se define a partir de su representación mediante vectores de características. A este fin, se establece una familia de transformaciones, $\vec{M}_k()$, que asocian a cada carga útil un vector de acuerdo al esquema que se detalla a continuación.

En primer lugar, se consideran todas las secuencias posibles de k de caracteres de un alfabeto de tamaño finito (ASCII en nuestro caso), que son ordenadas de acuerdo a un criterio arbitrario. Evidentemente, existirán n^k posibles secuencias, siendo n el tamaño del alfabeto considerado.

La transformación $\vec{M}_k(p)$ asocia a la carga útil p un vector de características de dimensión n^k tal que la componente i-ésima de dicho vector corresponde al número de veces que la cadena i-ésima aparece en dicha carga útil.

A partir de la parametrización descrita se define la distancia entre dos cargas útiles, p_1 y p_2, como la distancia euclídea entre sus correspondientes vectores de características:

$$D(p_1, p_2) = d(\vec{M}_k(p_1), \vec{M}_k(p_2)) \qquad (2)$$

Debido a la gran dimensionalidad de los vectores de características, el cálculo de la distancia entre ellos no es inmediato, pudiendo llegar a ser inviable en función de los valores de k considerados. A modo de ejemplo, considerando el código ASCII y $k = 5$ se obtendrían vectores de 1099511627776 componentes. Afortunadamente, la distancia puede evaluarse a partir de 3 productos escalares y, a su vez, el producto escalar considerado se puede interpretar como el número de subcadenas comunes que aparecen en ambos operandos. A partir de esta interpretación es posible aplicar técnicas basadas en programación dinámica para su evaluación directa a partir de las cargas útiles, lo que permite abordar computacionalmente el problema. Una descripción detallada del algoritmo puede encontrarse en [9], mientras que su aplicación en el IDS N3 se describe en [6].

2.2. Escenario de evaluación

En este trabajo se analiza la aplicación del sistema a instancias de tráfico del protocolo HTTP. Éste transporta elementos con una sintaxis y semántica bien establecidos, definidos en los correspondientes RFC. En concreto, las cargas útiles de HTTP responden a lo establecido en los RFC 2068 y 2396, correspondiendo a cadenas de caracteres.

Conjuntos de datos utilizados			
	Tipo	Descripción	N
$\mathcal{N}_{norm}$	Normal	Peticiones HTTP para entrenamiento	3262
$\mathcal{N}_{eval}$	Normal	Peticiones HTTP para evaluación	1397
$\mathcal{A}$	Ataques	Variantes de 86 ataques HTTP	119

Cuadro 1: Escenario experimental utilizado.

Para la evaluación del detector se utilizarán trazas de tráfico capturado de la red. En particular, se establecerá un escenario compuesto por tres conjuntos de datos, cada uno de ellos conteniendo cargas útiles del protocolo HTTP (véase el Cuadro 1). Los dos primeros conjuntos, que denominaremos $\mathcal{N}_{norm}$ y $\mathcal{N}_{eval}$, han sido extraídos de las trazas proporcionadas por el Programa de Evaluación de IDS de DARPA [11] correspondientes a las semanas 1 y 3, seleccionando al azar el 70 % de las muestras para $\mathcal{N}_{norm}$ y el 30 % restante para $\mathcal{N}_{eval}$. Estos conjuntos contienen tráfico normal y serán utilizados para el entrenamiento, es decir, como modelo de normalidad, y para la evaluación del rendimiento del sistema, respectivamente. Por otra parte, el tercero de los conjuntos, denominado $\mathcal{A}$, contiene tráfico correspondiente a ataques que ha sido generado simulando unas condiciones experimentales idénticas a las de los otros dos conjuntos. Para ello se ha recopilado la información sobre ataques a los servicios HTTP descrita en la base de datos arachNIDS [2] y se han implementado programas que los generan. Una descripción más detallada del escenario puede encontrarse en [6].

2.3. Complejidad computacional

La complejidad computacional del algoritmo de detección, de acuerdo a la Ec. (1), viene determinada por dos aspectos diferenciados: la evaluación de las distancias entre cargas útiles

y la búsqueda en el modelo de la distancia mínima, esto es, del vecino más cercano.

El algoritmo utilizado para la evaluación de las distancias presenta una complejidad computacional del orden de $O(l^2)$ operaciones, siendo l la longitud de las secuencias.

Por otra parte, la búsqueda del vecino más cercano requiere la evaluación de las distancias desde cualquier secuencia en el modelo de normalidad a la secuencia a clasificar, así como la comparación de los valores obtenidos a fin de encontrar el mínimo. Por tanto, si el modelo se compone de N secuencias, se requerirá evaluar N distancias y realizar $N-1$ comparaciones. Dado que la complejidad de la evaluación de las distancias es claramente superior a la de las comparaciones, podemos, en primera aproximación, despreciar estas últimas.

En consecuencia, el tiempo requerido para determinar si una carga útil dada es anómala o no puede aproximarse mediante la expresión:

$$t_{detec} \approx C \cdot \tau(l^2) \cdot N \qquad (3)$$

donde C es un factor constante.

En el escenario descrito, el tamaño del modelo es de 3262 cargas útiles. Por otra parte, el tiempo necesario para realizar el cálculo de las distancias ha sido evaluado durante los experimentos realizados con un procesador Pentium 4 a 2.4 GHz y 1 GB de RAM. Este tiempo oscila entre los 0,000298 y los 0,0579 ms, con un valor medio de 0,00483 ms. Por tanto, el tiempo medio necesario para determinar si una petición es normal o anómala será $t_{detec} \approx 0,00483 * 3262 = 15{,}76ms$, lo que en términos de rendimiento se traduciría en el procesamiento de 63 peticiones por segundo. Este resultado es claramente insuficiente, si bien es importante reseñar que no es inferior al de otros IDS actualmente operativos.

3. Aplicación de los algoritmos de agrupamiento

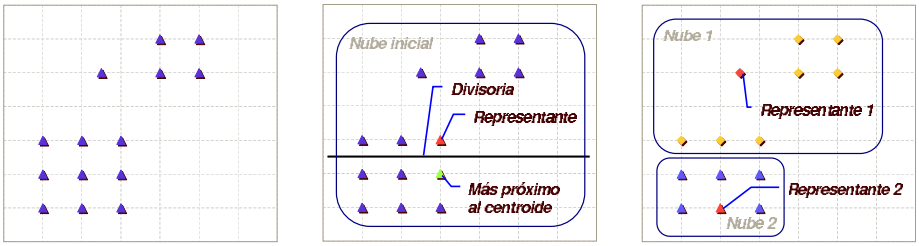
De la discusión anterior se deduce que resulta conveniente, de cara a su aplicación en un entorno real, reducir la complejidad computacional del IDS N3. A este fin, de acuerdo con la Ec. (3), se podrían aplicar técnicas que permitiesen reducir el tamaño del modelo, N. Evidentemente, esta reducción debe realizarse sin degradar la eficacia en la detección, es decir, el modelo debe seguir siendo representativo del tráfico considerado normal. El problema planteado se encuentra ampliamente tratado en la bibliografía, existiendo soluciones ampliamente aceptadas como las basadas en las técnicas de agrupamiento.

$\mathcal{N} \rightarrow$ modelo original ($\mathcal{N} = \{q\}$)
$\mathcal{N}' \rightarrow$ modelo reducido ($\mathcal{N}' = \{p_i\}$)
$n \rightarrow$ número de clases/representantes ($n = card(\mathcal{N}')$)
$C_i \rightarrow$ clase i
$n_i \rightarrow$ número de elementos en C_i
$p_i \rightarrow$ representante de la clase i
$p_i^n \rightarrow$ auxiliar de la clase i
$cl(q) \rightarrow$ clase a la que pertenece q
$D_m \rightarrow$ distorsión media
$D_m = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{n_i} \sum_{\forall q \in C_i} D(q, p_i)$

Cuadro 2: Notación y definiciones utilizadas.

En general, las técnicas de agrupamiento establecen una representación de un conjunto de elementos mediante la selección de un conjunto de *representantes*, de cardinalidad inferior. Para ello, mediante criterios de proximidad o similitud se establecen grupos de elementos (*clases* o *nubes*), cada una de las cuales tendrá un representante.

El aspecto más relevante de estos algoritmos reside en los mecanismos de selección de los miembros de cada clase y de sus representantes. Uno de los algoritmos más utilizados es el denominado *k-medias iterativo* [10], cuyo funcionamiento se esquematiza en el Algoritmo 1 (en el Cuadro 2 se describe la notación utilizada). Este algoritmo realiza sucesivas biparticiones de cada nube, reclasificando sus elementos en función de dos *atractores*, que determinan la pertenencia de un punto a una de las dos nubes resultantes de cada bipartición a partir de la determinación de la proximidad o similitud del punto considerado a cada uno de ellos. Los atractores considerados son el representante, obtenido como promedio de los vectores en la clase a dividir, y una versión perturbada del mismo.



a) b) c)

Figura 2: Ejemplo de operación del algoritmo de agrupamiento $C0$. a) Situación inicial. b) Selección del representante (centroide) y su versión perturbada. c) Nubes resultantes.

$C_1 = N$
Calcular D_m
Representante: $p_1 = promedio(q \in C_1)$
Mientras $\Delta D_m <$umbral-prefijado
  Para cada nube C_i
    Perturbación del representante:
      $p_i^n = p_i + \delta$
    División: $n++$; $C_i \Rightarrow C_i \cup C_n$
    Reclasificación:
      $\forall q \in C_i$: si $D(q, p_i) > D(q, p_i^n)$
        $cl(q) := C_n$
    Recálculo de representantes:
      $p_i = promedio(q \in C_i)$;
      $p_n = promedio(q \in C_n)$
  Nuevo modelo: $\mathcal{N}' = \{p_i\}$
  Evaluar D_m

Algoritmo 1: Algoritmo *k-medias*

3.1. Aplicabilidad de los algoritmos

Dos son las características del sistema N3 que dificultan la aplicación inmediata de los algoritmos de agrupamiento descritos en la bibliografía. En primer lugar, dado un vector de características, no existe necesariamente una única secuencia origen que lo genere, siendo posible, además, que no exista ninguna secuencia origen para dicho vector. En segundo lugar, las distancias se evalúan a partir de las cargas útiles, no a partir de los vectores de características, no disponiéndose de dichos vectores de forma explícita. De esta forma, se imposibilita el uso que cualquier algoritmo que obtenga los representantes de las clases a partir de promedios u operaciones similares.

En la bibliografía se describen algoritmos que, en lugar de obtener promedios, eligen como representante de una clase a alguno de los vectores de la misma. Tal es el caso del denominado *k-medoids* [8], que puede ser descrito como una variación del algoritmo k-medias con la salvedad de que se elige como representante al elemento de la clase más próximo al promedio, es decir, aquel cuya suma de distancias a los restantes miembros sea mínima. Sin embargo, en la bibliografía no se describe ninguna versión iterativa de dicho algoritmo, tal como ocurre con el k-medias. Por otra parte, k-medoids presenta dos inconvenientes [8] para su aplicación al problema considerado: no es posible determinar el número óptimo de clases y, en su formulación actual, presenta un muy elevado coste computacional.

A continuación se desarrolla una versión iterativa del algoritmo k-medoids, inspirada por el algoritmo k-medias iterativo.

3.2. Propuesta preliminar: algoritmo C0

En primera aproximación, puede implementarse una versión de k-medoids iterativa sin más que modificar el mecanismo de elección de los atractores usado en el algoritmo k-medias iterativo (Algoritmo 1). Así, los dos atractores se eligen, el primero, considerando como representante aquel cuya suma de distancias a los restantes miembros sea mínima; y, el segundo, el miembro de la clase más próximo al representante (véase la Fig. 2).

Inicialmente, k-medias iterativo y el algoritmo propuesto realizan una bipartición de todas las nubes en cada iteración. Para evitar los problemas asociados a la necesidad de dividir nubes con un único elemento, en lo que sigue únicamente se dividirá una nube en cada

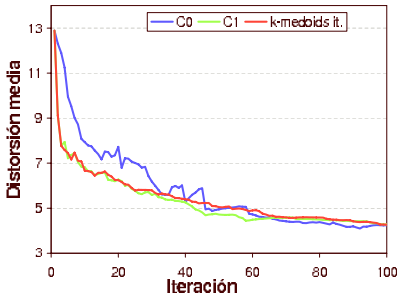


Figura 3: Evolución de la distorsión media en función de la iteración (número de representantes).

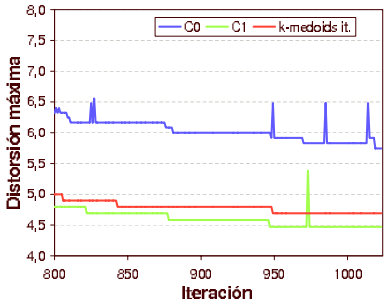


Figura 4: Evolución de la distorsión máxima en función de la iteración. Se muestra un detalle de las últimas iteraciones.

iteración. En principio, la nube seleccionada será la que presente mayor distorsión interna.

Por otra parte, es posible que tras la separación de una clase y posterior reclasificación y obtención de los nuevos representantes, existan puntos que se encuentren más próximos al representante de una clase adyacente que al de su propia clase (véase la Fig. 2.c). Para evitar este problema, se añade una etapa final en cada iteración que reclasifica todos los elementos, asociándolos a la clase cuyo representante se encuentre más próximo.

La aplicación del algoritmo con las modificaciones descritas, que denominaremos $C0$, presenta una evolución que se muestra en la Fig. 3. En esta curva resultan evidentes las fuertes oscilaciones en la evolución de la distorsión media, que indican un comportamiento inadecuado del mismo, especialmente en las primeras iteraciones. Un análisis de los mecanismos aplicados y un caso de estudio nos llevan a concluir que la bipartición de las nubes y la posterior selección de representantes no se realizan de forma óptima en el caso de que las nubes sean asimétricas (véase la Fig. 2).

3.3. Revisión del algoritmo: algoritmo $C1$

La partición subóptima de las nubes mencionada anteriormente se produce debido a que es necesario seleccionar elementos pertenecientes a las nubes, lo que condiciona la ubicación de los atractores. La bipartición de la nube se realiza utilizando como umbral de decisión la mediatriz de ambos puntos, por lo que el resultado depende de la selección inicial de dichos puntos (véase la Fig. 2).

Para solucionar este problema, proponemos introducir una nueva modificación de tal forma que los atractores sean el representante y el punto más distante a éste. Es posible definir el *radio* de la nube como la distancia entre los dos puntos mencionados.

Por otra parte, dado que la eficacia del IDS depende de las distancias entre los vectores normales y los anómalos, consideramos relevante que la bipartición se realice sobre la nube de mayor radio. De esta forma, es de esperar que la distancia máxima entre cualquier punto y su representante disminuirá, lo que debería evitar que se degrade la eficacia del detector.

El algoritmo resultante, denominado $C1$, presenta una evolución con mejor comportamiento que la del algoritmo $C0$ (Fig. 3), si bien la convergencia no es monótona. Por otra parte, si consideramos la evolución de la distorsión máxima, observamos un comportamiento altamente fluctuante con oscilaciones en todo el rango de iteraciones (Fig. 4).

Este comportamiento oscilante para la distorsión máxima puede tener implicaciones sobre la eficacia de la detección, ya que las capacidades del clasificador dependen de la diferencia entre la máxima distancia sobre las cargas útiles normales y la mínima distancia sobre las anómalas. Del análisis del comportamiento del algoritmo podemos deducir que existen situaciones en las que la bipartición continúa realizándose de forma subóptima (Fig. 5).

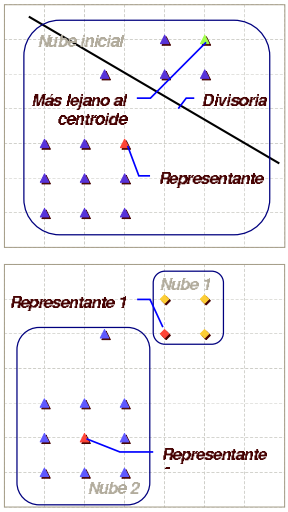


Figura 5: Ejemplo de operación del algoritmo de agrupamiento $C1$.

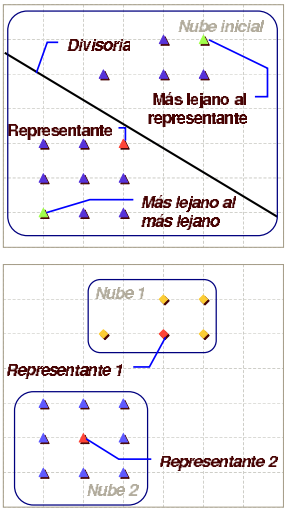


Figura 6: Ejemplo de operación del algoritmo de agrupamiento k-medoids iterativo propuesto.

3.4. Algoritmo k-medoids iterativo

A fin de evitar los problemas detectados en el algoritmo $C1$, se propone introducir una nueva modificación relacionada con la bipartición. A este fin, se considerarán como elementos atractores el punto más distante al representante y el punto más distante a este último. Un esquema del algoritmo se muestran en Algoritmo 2. Como se puede observar en la Fig. 6, este mecanismo realiza una división correcta de la nube considerada como caso de estudio.

La evolución de la distorsiones media presenta un comportamiento muy similar a la del algoritmo $C1$ (Fig. 3), aunque las inestabilidades son menores. Por otra parte, el comportamiento respecto de la distorsión máxima es claramente superior al de los otros algoritmos analizados (Fig. 4).

4. Resultados experimentales

Las tres variantes propuestas han sido aplicadas al modelo $\mathcal{N}_{norm}$, mostrándose los resultados relativos a la detección en la Fig. 7. Se puede comprobar que el algoritmo $C0$ (curvas azules) presenta unos pobres resultados, degradándose rápidamente la capacidad de detección al reducir el tamaño del modelo (de

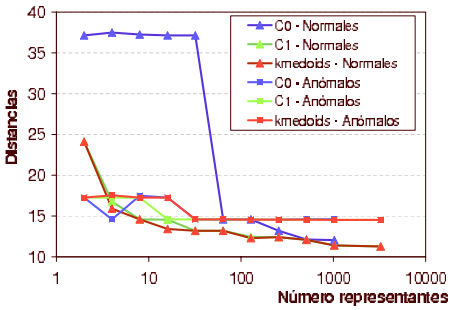


Figura 7: Evolución de las distancias con el número de representantes.

derecha a izquierda en la figura). Por el contrario, los algoritmos $C1$ y k-medoids iterativo (curvas verdes y rojas, respectivamente) presentan comportamientos que permiten reducir en varios órdenes de magnitud el tamaño del modelo. En concreto, ambos operan correctamente para $N = 4$, si bien las distancias entre los umbrales de decisión son mayores en el caso de k-medoids iterativo.

Con las reducciones en el tamaño del modelo obtenidas, considerando, por ejemplo $N = 25$ prototipos, que aún proporcionan resultados de detección satisfactorios, y utilizando los tiempos de cómputo indicados en el Apartado 2, se consigue una reducción del tiempo de detección del valor original $t_{detec} = 15{,}76ms$

$C_1 = N$
Calcular D_m
Función selección representante:

$$repr(C_i) = arg \min_{\forall q \in C_i} \sum_{\forall p \in C_i, p \neq q} D(q,p)$$

Representante: $p_1 = repr(C_1)$
Mientras $\Delta D_m <$umbral-prefijado
  Eliminar representantes $\mathcal{N}' = \emptyset$
  Para cada nube C_i
    Obtener el representante: $p_i = repr(C_i)$
    Obtener más lejano al representante:
      $p_i^n = arg\,\text{máx}_{\forall q \in C_i}\{d(p_i, q)\}$
    Evaluar el *radio* de la nube:
      $R = D(p_i, p_i^n)$
    Añadir representante: $\mathcal{N}' = \mathcal{N}' \{p_i\}$
  Para la nube C_r con mayor R:
    Eliminar representante: $\mathcal{N}' = \mathcal{N}' - \{p_r\}$
    Obtener más lejano al más lejano:
      $p_r^f = argmax\{d(p_r^n, q)\} \forall q \in C_r$
    División: $n++$; $C_r \Rightarrow C_r \cup C_n$
    Reclasificación:
      $\forall q \in C_r$: si $D(q, p_r^f) > D(q, p_r^n)$
        $cl(q) := C_n$
    Recálculo de representantes:
      $p_r = repr(C_r)$; $p_n = repr(C_n)$
    Actualizar modelo: $\mathcal{N}' = \mathcal{N}' \cup \{p_r, p_n\}$
  Evaluar D_m

Algoritmo 2: Algoritmo *k-medoids* iterativo.

a un valor de $t_{detec} = 0{,}121ms$. Con estos valores, el detector podría gestionar 8264 peticiones por segundo, obteniéndose un incremento significativo sobre el modelo original.

5. Conclusiones y trabajo futuro

Se ha mostrado que, mediante la aplicación de algoritmos de agrupamiento adecuados, es posible reducir en varios órdenes de magnitud el tamaño de los modelos utilizados por el IDS N3 sin degradar la eficacia de la detección. Esta reducción redunda en una mayor aplicabilidad del sistema, al disminuir los tiempos de procesamiento empleados por el detector.

Por otra parte, es necesario profundizar en algunos aspectos del algoritmo de agrupamiento propuesto, especialmente en lo que respecta a su validación y a la determinación automática del tamaño óptimo del modelo.

Referencias

[1] Allen, J. et al.; "State of the Practice of Intrusion Detection Technologies". Technical Report CMU/SEI-99-TR-028, Software Engineering Institute, Carnegie Mellon Univ., 2000.

[2] *Arachnids: Advanced Reference Archive of Current Heuristics for Network Intrusion Detection Systems.* http://www.whitehats.com/ids, 2003.

[3] Athanasiades, N. et al.; *Intrusion Detection Testing and Benchmarking Methodologies*, Proc. IWIA'03, Darmstadt (Germany), 2003, pp. 63-72.

[4] Axelsson, S.; *Intrusion Detection Systems: A Survey and Taxonomy.* Tech. Report 99-15, Dept. of Computer Engineering, Chalmers Univ. of Technology, 2000.

[5] Estévez-Tapiador, J.M. et al.; *NSDF: a computer network system description framework and its application to network security*, Computer Networks 43, pp. 573-600, 2003.

[6] Estévez Tapiador, J.M.; *Detección de intrusiones en redes basada en anomalías mediante técnicas de modelado de protocolos*, Tesis doctoral, Univ. Granada, 2004.

[7] Estévez-Tapiador, J.M. et al.; *N3: A geometrical approach for network intrusion detection at the application layer*, ICCSA 2004, LNCS 3043, pp. 841-850, 2004.

[8] Estivill-Castro, V. et al. *Robust distance-based clustering with applications to spatial data mining.* Algorithmica 30(2):216–242, 2001.

[9] Gusfield, D.; *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, ISBN: 0521585198, Cambridge Univ. Press, 1997.

[10] Linde, A., Buzo, Y., Gray, R.M., *An Algorithm for Vector Quantizer Design.* IEEE Trans. on Communications, COM-28(1), pp. 84-95, 1980.

[11] Lippmann, R. et al.; *The 1999 DARPA Off-line Intrusion Detection Evaluation*, Computer Networks, Vol. 34, No. 4, pp. 579-595, 2000.