Una Aproximación Efectiva a la Detección de Anomalías en el Tráfico TCP/IP Usando Técnicas de Inteligencia Artificial

Jorge Couchet[1], Miriam Steiner[2], Rodrigo San Vicente[3], Enrique Ferreira[4]

[1]: Shell Corporation, Uruguay. jorge.couchet@shell.com
[2]: Impresiones y Publicaciones Oficiales, Uruguay. msteiner@impo.com.uy
[3]: Administración Nacional de Telecomunicaciones, Uruguay. rsvm@adinet.com.uy
[4]: Universidad Católica del Uruguay. enferrei@ucu.edu.uy

Resumen

El presente artículo introduce una aproximación al problema de "*Anomaly Intrusion Detection*" basada en una combinación de algoritmos de "*Machine Learning*" (ML) supervisados y no supervisados. Los objetivos que se persiguen son: el modelar en forma efectiva el tráfico de una organización y el reducir en forma substancial el porcentaje de Falsos Positivos mientras se mantiene un nivel razonable de detección de anomalías. Se presenta una arquitectura basada en un conjunto de "*Self-Organizing Maps*" (SOM) para el modelado del tráfico y en el uso de "*Linear Vector Quantization*" (LVQ) para la clasificación definitiva de los paquetes de tráfico. Los algoritmos desarrollados usan Snort para el pre-procesamiento del tráfico de red, y están pensados para ser un complemento de esta herramienta. Los resultados alcanzados hasta el momento muestran que se pueden lograr niveles aceptables de acierto en comparación con otras técnicas. Al final se plantean las conclusiones extraídas del trabajo y direcciones en las cuales se puede continuar el desarrollo y mejorar los resultados obtenidos.

Palabras claves:
Inteligencia Artificial, Machine Learning, Neural Nets, Computacional Intelligence, Learning Vector Quantization, Self-Organizing Maps, Anomaly Intrusion Detection.

1 Introducción

1.1 Seguridad de la información

Las Tecnologías de Información (TI's) hoy en día son una realidad cada vez más frecuente en las empresas, sean estas pequeñas, medianas o grandes. Prácticamente no quedan organizaciones que no dependan de éstas como infraestructura vital para la realización de sus actividades. Esta fuerte dependencia en las TI's entraña sus riesgos, entre ellos, la interrupción de los servicios en la plataforma de TI puede ocasionar severos trastornos, poniendo en juego activos de la empresa físicos e intangibles, como ser clientes, imagen institucional, oportunidades de negocio, etc.

Las fuentes que pueden afectar la estabilidad de la infraestructura de TI son varias, pudiéndose destacar las siguientes: a) problemas relacionados con el "*Hardware*", b) problemas en las distintas aplicaciones, c) falta de entrenamiento adecuado en el personal, y d) problemas con la ***Seguridad Informática***.

La preponderancia de los riesgos asociados con la *Seguridad Informática* como fuente de inestabilidad de los sistemas de información ha estado creciendo vertiginosamente en los últimos años, y por consiguiente los gastos de las empresas en soluciones tecnológicas para la mitigación del riesgo tienen una evolución similar. Usando fuentes del CERT [1], el crecimiento de los riesgos asociados a la Seguridad Informática se puede observar claramente en los siguientes cuadros:

Número de incidentes de seguridad reportados anualmente:

Año	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
Incidentes	252	406	773	1,334	2,340	2,412	2,573	2,134	3,734	9,859

Año	2000	2001	2002	2003
Incidentes	21,756	52,658	82,094	137,529

El explosivo crecimiento de los incidentes de seguridad, vuelve imperiosa la necesidad de desarrollar técnicas eficientes de detección de intrusos. Hay dos aproximaciones generales utilizadas para el desarrollo de "*Intrusion Detection Systems*" (IDS):

"Misuse Detection": Opera con patrones preparados de antemano ("*Signatures*"), siendo los mismos configurados para reconocer ataques basados en alguna vulnerabilidad conocida

"Anomaly Detection": Utilizando algoritmos de "*Machine Learning*" [2] se implementa un modelo de la actividad normal de los sistemas de una organización dada, y cualquier desviación del mismo es considerada anormal o sospechosa.

La mayoría de los IDS disponibles hoy en día son del tipo "*Misuse Detection*", debido a que es la tecnología más sencilla de implementar, sin embargo adolecen de los siguientes problemas, lo cuales les restan en gran medida su eficacia: a) son extremadamente rígidos, pudiendo detectar únicamente aquellos ataques de los cuales se dispone su "*Signature*", b) es necesario actualizar constantemente la base de datos de "*Signatures*", las cuales son desarrolladas en forma manual cuando un ataque se vuelve conocido (y por tanto ya tuvo la

ocasión de dañar un número importante de sistemas), y c) un intruso con un suficiente conocimiento de las "*Signatures*" puede modificar ligeramente su ataque de forma de evadirlas, pasando inadvertido al IDS

Los problemas anteriores se pueden resolver en gran parte con IDS basados en "*Anomaly Detection"*, los cuales tienen la ventaja de adaptarse en forma dinámica y automática a las características relevantes de la actividad de TI de una organización, por lo que no sería necesario el conocer los ataques de antemano para su detección, mejorándose la capacidad de respuesta ante los riesgos derivados de la *Seguridad Informática*. Debido a lo anterior muchos de los esfuerzos actuales de investigación en el área de *Intrusion Detection* están enfocados en esta dirección, como puede observarse en por ejemplo: [3], [4], [5], [6], [7], y [8].

El presente trabajo presenta una nueva arquitectura basada en SOM y LVQ para atacar el problema de Anomaly Detection. Se describen primero brevemente las arquitecturas SOM y LVQ y luego como son usadas en el sistema propuesto. Se muestran los resultados de una implementación del sistema basada en Snort [9] y se comparan con otras arquitecturas. En la sección conclusiones se discuten dichas comparaciones y se mencionan las líneas de investigación futuras.

1.2 Self-Organizing Maps (SOM)

La SOM [10] es una red neuronal con un aprendizaje no supervisado competitivo, que realiza una transformación de un espacio multidimensional en una serie de neuronas de tal forma que las distancias relativas entre los puntos del espacio de entrada se conservan. Las neuronas suelen estar dispuestas en un plano (2-D grid).

Cada neurona **i** de la SOM tiene asociado un vector de pesos $\mathbf{m_i} = [\mathbf{m_{i1}},...,\mathbf{m_{in}}]^T$, donde n es la dimensión del espacio de entrada. Las neuronas de una red SOM están relacionadas con las neuronas de su entorno mediante una relación de vecindad. Generalmente, la estructura de la red es rectangular o hexagonal. La estructura determina la vecindad. Las neuronas adyacentes a una dada forman la vecindad próxima, para una neurona **i** esta vecindad se representa por **Ni**. En la forma básica de la red SOM, las relaciones topológicas y el número de neuronas son fijos desde el principio. El número de neuronas determina la granularidad de la transformación, lo que afecta a la exactitud y capacidad de generalización de la red.

Durante el proceso iterativo que es el entrenamiento, la SOM debe descubrir por sí misma rasgos comunes, regularidades, correlaciones o categorías en los datos de entrada, e incorporarlos a su estructura interna de conexiones. Se dice, por tanto, que las neuronas deben auto organizarse en función de los estímulos procedentes del exterior. La red tiende a aproximar la densidad de probabilidad de los datos. Los vectores de pesos tienden hacia donde hay muchos datos, de manera que donde haya pocos datos habrá pocos vectores de pesos. En cada paso de entrenamiento, se toma aleatoriamente un vector **x** del espacio de entrada. Se calculan las distancias entre todos los vectores de peso y el vector del espacio de entrada que se ha tomado. La neurona ganadora es aquella para la que la distancia entre su vector de pesos y el vector de entrada elegido es mínima, cumpliendo con la siguiente ecuación[1]:

[1] Donde ||.|| hace referencia a la función distancia, generalmente la Euclídea.

$$c = \arg\min_{i}\{\| x - m_i \|\}$$

Después de encontrar la neurona ganadora se actualizan los vectores de pesos. La neurona ganadora y su vecindad son movidas hacia el vector de entrada en el espacio de entradas. La regla que se sigue para actualizar los pesos de la neurona **i** viene dada por la siguiente ecuación:

$$m_i(t+1) = \begin{cases} m_i(t) + \alpha(t)[x(t) - m_i(t)], & \mathrm{i} \in \mathrm{N_c}(t) \\ \mathrm{m_i}(t), & \mathrm{i} \notin \mathrm{N_c}(t) \end{cases}$$

donde **t** denota el tiempo. El vector **x(t)** es el vector de entrada escogido aleatoriamente de entre los datos de entrada en el instante **t**, $\mathbf{N_c(t)}$ es la función de vecindad[2] de la neurona ganadora **c** y no cambia con el tiempo. El coeficiente **α(t)** es el tasa de aprendizaje, que está acotado entre 0 y 1 y decrece con el tiempo. El entrenamiento se hace en dos fases. En la primera fase se usan valores grandes de **α** (0.3 hasta 0.99), mientras que en la segunda fase se usan valores pequeños (0.01 hasta 0.1). Si la función de vecindad no es constante, en la primera fase contempla vecindades mayores que en la segunda.

1.3 Learning Vector Quantization (LVQ)

Learning Vector Quantization (LVQ) es un precursor de las redes SOM, y como las mismas puede considerarse una clase de Redes Neuronales. Una red LVQ está compuesta por dos capas: una de entrada, y otra de salida. Esta representa un conjunto de vectores de referencia, donde las coordenadas de los mismos son los pesos de las conexiones entre los vectores de entrada y las neuronas de salida.

Hay varios algoritmos LVQ: *LVQ1*, *LVQ2.1*, *LVQ3* y *OLVQ1*. A continuación se explica *LVQ1* que es el usado en el presente trabajo: un determinado número de "*codebook vectors*" $\mathbf{m_i}$ son puestos en el espacio de entrada para aproximar los distintos dominios del vector de entrada **x** por sus valores quantizados. Usualmente varios "*codebook vectors*" son asignados a cada clase del espacio de entrada de los vectores **x**, decidiéndose que **x** pertenece a la misma clase del $\mathbf{m_i}$ más cercano a él: $\mathbf{m_c}$, el cuál cumple la siguiente ecuación:

$$m_c = \arg\min_{i}\{\| x - m_i \|\}$$

Los "codebook vectors" $\mathbf{m_i}$ pueden encontrarse como los valores asintóticos del siguiente proceso de aprendizaje: sea x(t) una muestra del espacio de entrada, y sea $\mathbf{m_i(t)}$ la secuencia de los $\mathbf{m_i}$ en un dominio discreto del tiempo. Comenzando con valores iniciales debidamente seleccionados, las siguientes ecuaciones definen el proceso básico del aprendizaje *LVQ1*:

$$m_c(t+1) = \begin{array}{l} \mathrm{m_c}(t) + \alpha(\mathrm{t})[\mathrm{x(t)} - \mathrm{m_c(t)}] \\ \text{si x y } \mathrm{m_c} \text{ pertenecen a la misma clase} \end{array}$$

$$m_c(t+1) = \begin{array}{l} \mathrm{m_c}(t) - \alpha(\mathrm{t})[\mathrm{x(t)} - \mathrm{m_c(t)}] \\ \text{si x y } \mathrm{m_c} \text{ pertenecen a clases distintas} \end{array}$$

$$m_i(t+1) = \mathrm{m_i}(t) \quad \text{para } \mathrm{i} \neq \mathrm{c}$$

donde $0 < \mathbf{\alpha(t)} < 1$, y $\mathbf{\alpha(t)}$ puede ser constante o decrecer monótonamente con el tiempo.

[2] Dos de las funciones de vecindario más usadas son: a) Bubble y b) Gaussian.

2 Detección de anomalías basadas en SOM / LVQ

El problema que se trata de abordar en este trabajo es la clasificación de los paquetes de tráfico TCP que llegan a un sistema en "normales" o de "ataque". La arquitectura está basada en los trabajos de Lichodzijewski et al. [11], [12] y de Kayac1k [13], usándose de los mismos los conceptos de una jerarquía de SOM's, la representación implícita del tiempo mediante una ventana temporal de muestras, y el uso de una tercera capa para la resolución de casos más complejos clasificados como "indefinidos" en una primera etapa. Los aportes nuevos del presente trabajo son la selección de un conjunto más amplio de atributos, la codificación de los mismos, la introducción de LVQ's para la reducción de Falsos Positivos, la forma de codificar las neuronas más representativas del primer nivel, y la definición especial de los casos indefinidos. Para reducir la complejidad de los experimentos se analiza la información al nivel de los paquetes del tráfico TCP (sin hacer uso de la información del nivel de conexión, o de otros protocolos).

2.1 Arquitectura general

La arquitectura general del sistema SOM/LVQ es como se muestra en el diagrama siguiente:

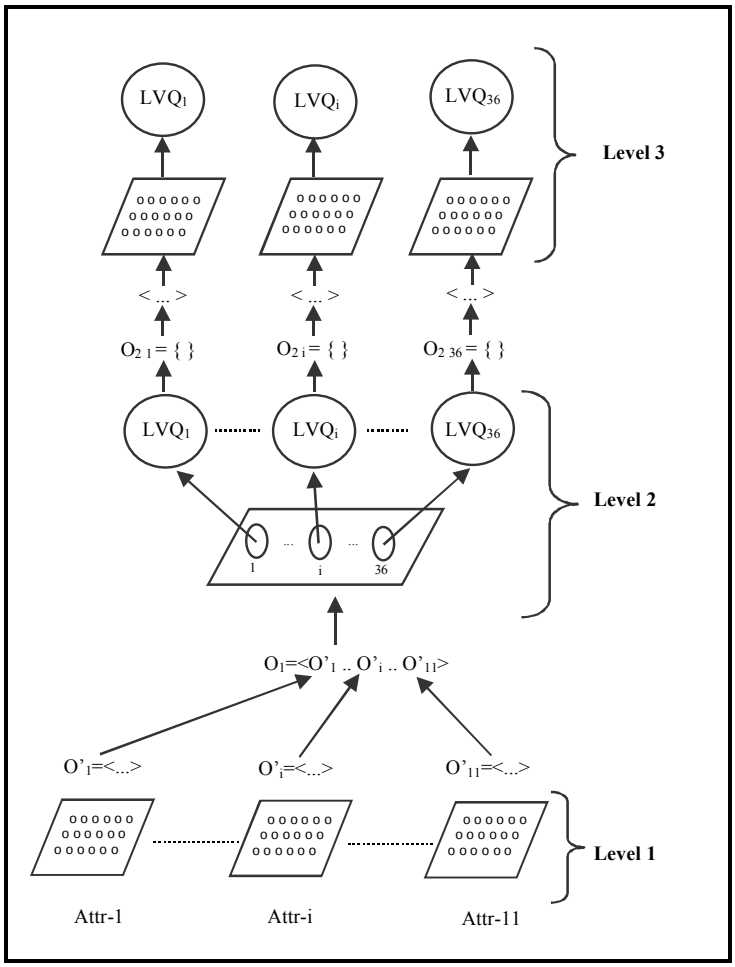

Figura 1

La misma está compuesta por tres niveles o capas, los cuales se detallan a continuación.

2.2 Primer nivel - "Clustering" de características

El primer nivel de la arquitectura está compuesto de 11 SOM's (6x6), una por cada atributo seleccionado para caracterizar el tráfico TCP de la organización. La entrada de cada SOM es un vector de dimensión 20, el cual mapea una ventana discreta de tiempo del mismo tamaño, cada uno de los elementos del vector es una muestra del atributo tomada en la sección correspondiente de la ventana [12].

El objetivo de este nivel es modelar las características sobresalientes de los atributos seleccionados, para obtener así un modelo del tráfico. Una vez entrenada cada una de estas SOM, se seleccionan de las mismas 6 neuronas (centros) con el fin de disminuir la dimensionalidad de la entrada al segundo nivel. Los criterios utilizados fueron dos: a) uso de una *Potential Function* [13], y b) la selección de tres neuronas representativas de tráfico normal, y tres representativas de tráfico de ataque.

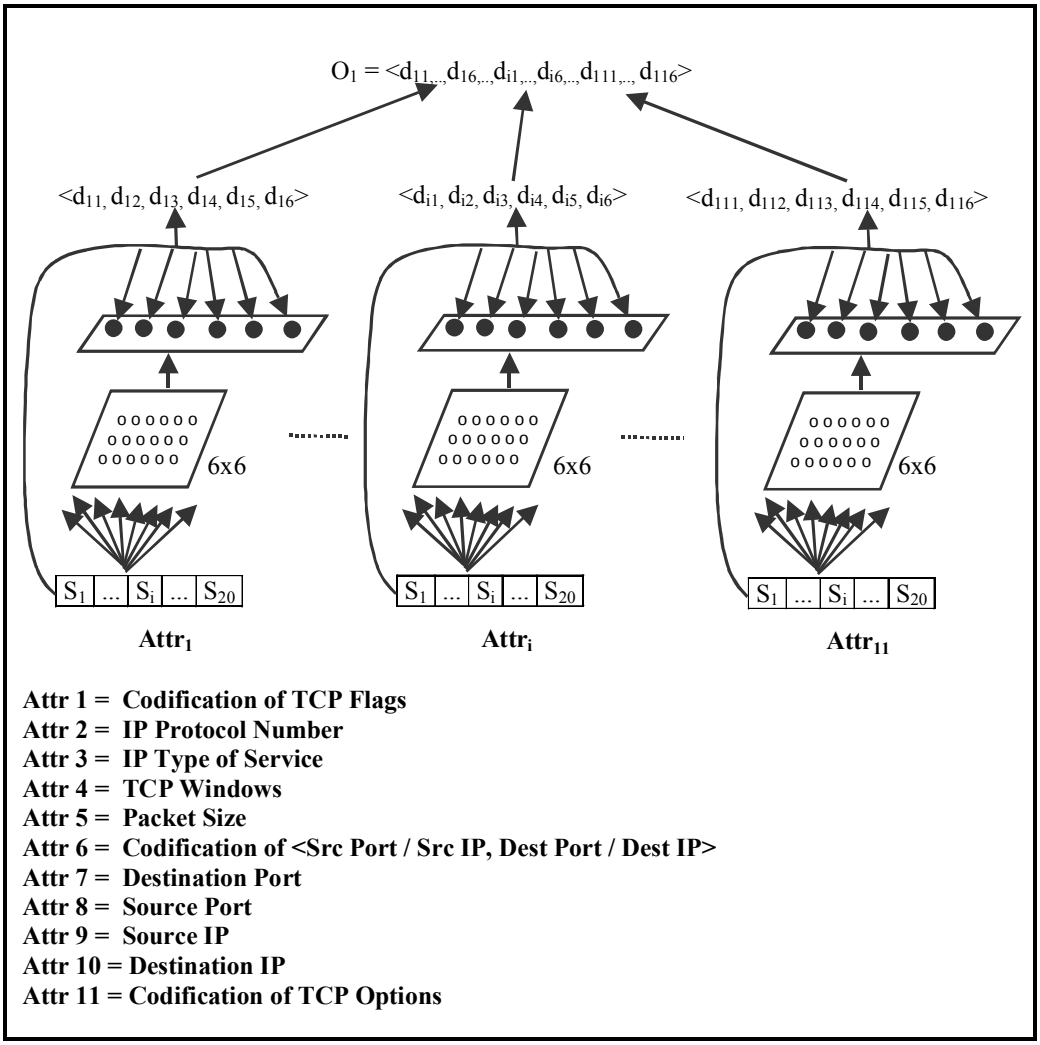

Figura 2

2.3 Segundo nivel - Agregación y clasificación

Este nivel está formado por una SOM (6x6), y un LVQ asociado a cada una de las neuronas de la misma (36 en total). La entrada de la SOM de segundo nivel es el vector conformado con las distancias de cada uno de los vectores de entrada de las SOM del primer nivel a las 6 neuronas seleccionadas de las mismas, por lo que la dimensión del espacio de entrada de la segunda capa es: 6x 11 = 66. El objetivo de esta SOM es capturar las relaciones de los distintos atributos a lo largo del tiempo, de forma de modelar efectivamente el tráfico de la organización.

Una vez realizado el aprendizaje de la SOM, se entrena en forma supervisada cada uno de los LVQ asociados a sus neuronas, con el subconjunto de vectores del espacio de entrada para los cuales la neurona del LVQ resulta ganadora. La etiqueta usada para el entrenamiento del LVQ es *MyLabel*, la cual se construye como:

$\frac{\text{Number Packet Attacks}}{\text{Number Packets}}$ en función de los paquetes que componen la ventana discreta de tiempo.

Es posible entrenar a los LVQ's usando uno de los siguientes mecanismos:
a) usando los valores de *MyLabel* tal cual fueron generados, y b) discretizando los valores de *MyLabel* en *Normal*, *Ataque* e *Indefinido*, donde 0 < *Indefinido* < *Umbral_Ataque*[3]. En este caso el sistema está realizando a este nivel una clasificación en 3 clases, dejándose para el siguiente nivel la clasificación final de los paquetes que quedaron en la clase *Indefinido*.

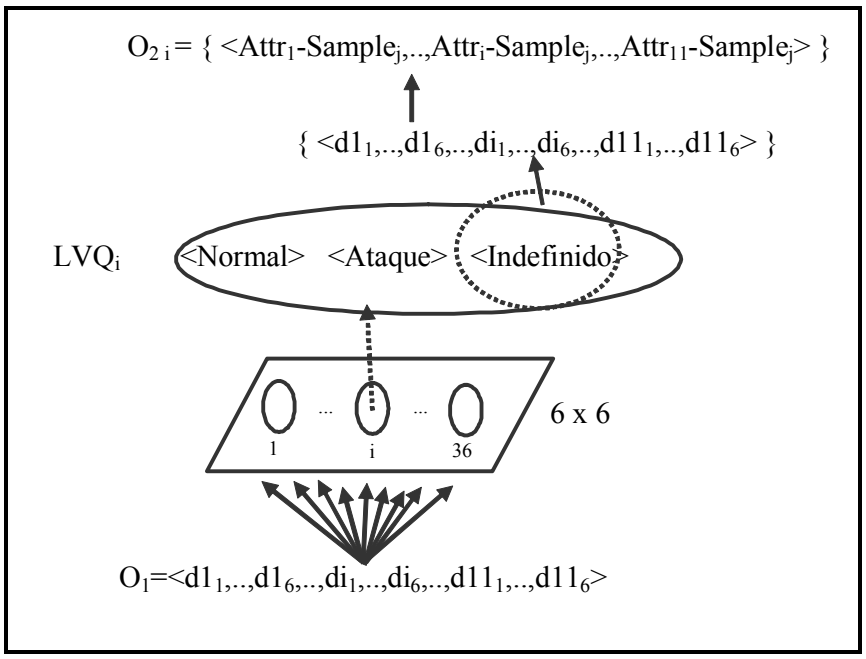

Figura 3

[3] Por ejemplo Umbral_Ataque puede ser 30%.

2.4 Tercer nivel - Procesamiento de indefinidos

El objetivo de este nivel es analizar aquellas ventanas donde el número de paquetes de ataques es muy bajo (menor que Umbral_Ataque), para eso cada uno de los LVQ's del segundo nivel tiene asignado una SOM (10 x 10), la cual se entrena con los paquetes pertenecientes a las ventanas asociadas a los prototipos *Indefinido* del LVQ asociado. Cada una de esas SOM's tiene asignada un LVQ para entrenar en forma supervisada los paquetes correspondientes, para lo cual se construye un vector con las distancias de estos a cada una de las neuronas de la SOM.

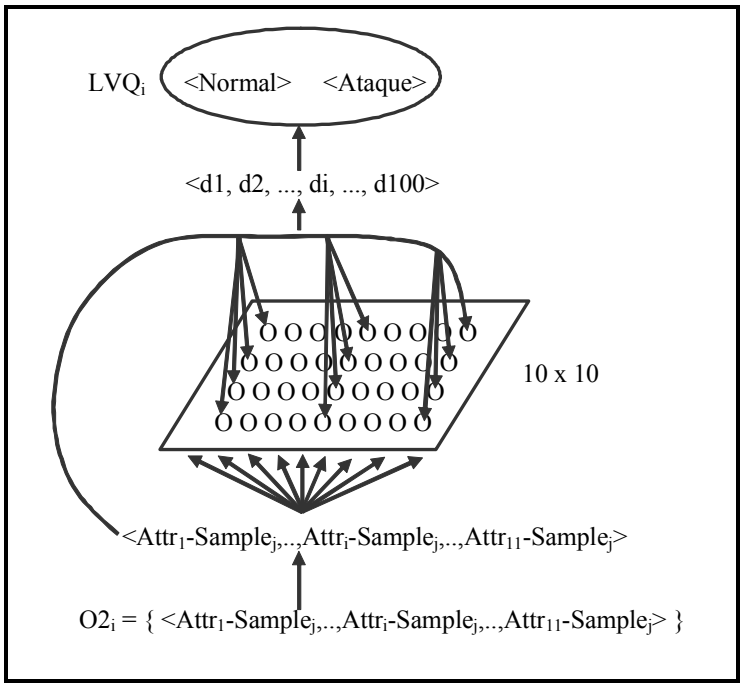

Figura 4

3 Experimentos

3.1 Diseño de los experimentos

3.1.1 Medidas de performance

Para evaluar la performance del sistema se consideran las siguientes medidas:

$$\text{Detection Rate} = \frac{\text{Number of Network Attacks Detected}}{\text{Number of Network Attacks}}$$

$$\text{False Positive Rate} = \frac{\text{Number of False Positives}}{\text{Total Number of Normal Connections}}$$

3.1.2 Conjunto de datos

Para los experimentos se usaron los datos del "*DARPA 1998 Intrusion Detection Problem*" [14], armándose en función de los mismos dos conjuntos uno de *Entrenamiento* con **1,514,848** registros, y otro de *Testeo* con **765,029** registros. Los resultados que se muestran en los experimentos son sobre el conjunto de *Testeo*.

3.2 Resultados de los experimentos

3.2.1 Preprocesamiento de los datos

Los datos de DARPA fueron preprocesados usando el Snort, de forma de extraer y codificar los atributos relevantes para modelar el tráfico TCP, solamente se usó información del nivel de los paquetes. Para realizar esta tarea se desarrolló un preprocesador de Snort, el cual es también capaz de brindar información de otros niveles (tal como atributos de la conexión TCP), así como de otros protocolos (UDP e ICMP por ejemplo).

3.2.2 Herramientas

La herramienta usada para implementar las redes SOM y LVQ fue una modificación de SOM_PAK [15].

3.2.3 Selección de características

Los atributos utilizados son los que se indican en la *Figura 2*, los mismos fueron seleccionados en función a su relevancia en el protocolo TCP, y por tanto por su capacidad para modelar dicho tráfico. Aquellos atributos que pueden a llegar a tener más de un valor en el mismo paquete (por ej. las TCP *Flags*) fueron codificados en un atributo monovaluado para no aumentar la dimensionalidad del problema.

3.2.4 Experimentos

Los experimentos fueron hechos en una computadora tipo PC pentium IV con 512MB de memoria corriendo Linux donde se instaló Snort y SOMPAK adaptados para este trabajo e implementó la arquitectura descripta anteriormente. El entrenamiento de la arquitectura llevó del orden de algunas horas para cada caso.

Para las tablas que muestran los resultados se usa la siguiente notación: ***Clasificación*** es lo que efectivamente el sistema SOM/LVQ clasificó, ***OK*** representa cuánto de lo clasificado en cada categoría es correcto, y ***Desviaciones*** indica cuánto de lo clasificado es incorrecto. Se utiliza las iniciales **N**, **A**, e **I** para indicar las 3 clases posibles: Normal, Ataque e Indefinido respectivamente. En el caso de las desviaciones se indica por ejemplo ***N-I*** para el caso de paquetes de clase ***Normal*** clasificados como ***Indefinidos***).

Selección de los 6 centros del primer nivel mediante Potencial Function y entrenamiento de los LVQ's con MyLabel sin discretizar:

	Clasificación			OK			Desviaciones					
	N	I	A	N	I	A	N-I	N-A	I-N	I-A	A-N	A-I
Nivel2	396674	132355	236000	227250	7657	208396	41845	14382	28584	13222	140840	82853
Nivel3	522269	0	71111	481170	0	50926	0	20185	0	0	41099	0

Selección de los 6 centros del primer nivel mediante Potencial Function y entrenamiento de los LVQ's con MyLabel discretizada:

	Clasificación			OK			Desviaciones					
	N	I	A	N	I	A	N-I	N-A	I-N	I-A	A-N	A-I
Nivel2	648290	66580	50159	239197	11238	33256	36754	7526	28848	9377	380245	18588
Nivel3	859386	0	429934	701701	0	112103	0	317831	0	0	157685	0

Selección de los 6 centros del primer nivel según: 3 Ataque / 3 Normales y entrenamiento de los LVQ's con MyLabel discretizada:

	Clasificación			OK			Desviaciones					
	N	I	A	N	I	A	N-I	N-A	I-N	I-A	A-N	A-I
Nivel2	413132	44929	306968	272431	3922	301335	6591	4455	44363	1178	96338	34416
Nivel3	637367	0	261213	246029	0	233798	0	27415	0	0	391338	0

Se puede observar claramente que los mejores resultados se obtienen cuando los centros del primer nivel son seleccionados en función de su representatividad respecto de las clases que se desea clasificar (en este caso 3 centros representando ataques, y 3 normales). En este caso la performance del sistema fue:

$$\text{Detection Rate} = \frac{301335 + 233798/20}{301335 + 96338 + 34416} \cong \frac{313025}{432089} \cong 72\%$$

$$\textit{False}\ \text{Positive Rate} = \frac{4455 + 27415/20}{272431 + 6591 + 4455} \cong \frac{5826}{283477} \cong 2.1\%$$

4 Discusión y Conclusiones

El objetivo del presente proyecto era desarrollar un método de "*Anomaly Detection"* con un "*Detection Rate"* razonable, y un "*False Positive Rate"* muy bajo, de forma de capturar la mayor cantidad de ataques, mientras que las alertas generadas fueran lo suficientemente confiables para un ambiente de producción. Los resultados obtenidos son prometedores en ese sentido, y animan a seguir profundizando la línea de investigación.

Algunos de los resultados obtenidos en proyectos de "*Anomaly Detection*" usando técnicas de "*Machine Learning*" se pueden observar en la siguiente tabla [13],[16]:

Técnica	Detection Rate	False Positive Rate
Clustering	93%	10%
K-NN	91%	8%
SVM	98%	10%
SOM	90%	7.6%

Se puede observar que el "*Detection Rate*" alcanzado en el presente proyecto es inferior al de los presentados en la tabla anterior, debiéndose a la combinación de los siguientes tres motivos:

- Una de las decisiones de diseño fue sacrificar un porcentaje del "*Detection Rate*" para mejorar el "*False Positive Rate*", el cuál es significativamente mejor que el de los otros proyectos comparados
- Un porcentaje significativo de los ataques presentes en el conjunto de datos de entrenamiento y validación son del tipo U2R ("*User to Root*"), los cuales son muy difíciles de modelar usando únicamente las características del protocolo TCP/IP
- En el estado actual del prototipo, la clasificación se está realizando únicamente a nivel de paquete, si se tiene en cuenta que lo que importa clasificar es una conexión, y que la misma está formada en promedio por cientos de paquetes, se puede observar que muy probablemente el "*Detection Rate*" mejore significativamente cuando se agregue el nivel de conexión

El resultado alcanzado si bien es positivo, está dentro de los objetivos del proyecto, es decir el desarrollo de un prototipo, el mismo en su estado actual es claramente inmaduro para su aplicación efectiva en ambientes de producción, por lo que se destaca la necesidad de continuar investigando de forma de alcanzar el nivel suficiente de madurez, para su aceptación por parte de las organizaciones del medio.

Los próximos pasos a seguir para mejorar el "*Detection Rate"* serán integrar al sistema información del nivel de *Conexión* y de *Host*, y el soporte para los protocolos ICMP y UDP. También se planea investigar el uso de las *Support Vector Machines* en el nivel de las LVQ's, y *Fuzzy Logic* para el pasaje de la información entre los niveles 1 y 2 de la arquitectura desarrollada.

5 Referencias

1. CERT Coordination Center (CERT/CC). *CERT/CC Statistics 1988-2005*. http://www.cert.org/stats/

2. Tom M. Mitchell. *Machine Learning*, McGraw-Hill, 1997.

3. M. Markou and S. Singh. *Novelty Detection:* A Review, Part II: Neural Network Based Approaches, 2003. http://www.dcs.ex.ac.uk/research/pann/master/2003.htm

4. Matthew Vincent Mahoney. *A Machine Learning Approach to Detecting Attacks by Identifying Anomalies in Network Traffic*. Doctor of Philosophy's thesis, Florida Institute of Technology, 2003. http://www.cs.fit.edu/~pkc/theses/mahoney03.pdf

5. Manikantan Ramadas. *Detecting Anomalous Network Traffic with Self-Organizing Maps*. Master's thesis, Ohio University, 2002.
http://zen.ece.ohiou.edu/~inbounds/DOCS/inbounds/mramadas-thesis.pdf

6. Stefano Zanero. *Un Sistema di Intrusion Detection Basato su Tecniche di Apprendimento non Supervisionato*. Bachelor's thesis, Politecnico di Milano, 2002.
http://www.elet.polimi.it/upload/zanero/papers/IDS-tesi.pdf

7. Dit-Yan Yeung and Calvin Chow. *Parzen-window network intrusion detectors*. In *Proc. of the Sixteenth International Conference on Pattern Recognition*, Aug 2002.

8. Brandon Rhodes, James A.Mahaffey, and James D. Cannady. *Multiple Self-Organizing Maps for Intrusion Detection*. In *Proceedings of the 23rd National Information Systems Security Conference*, 2000.

9. Snort. IDS "Open Source". http://www.snort.org

10. T.Kohonen. *Self Organizing Maps*. Springer, Third Extended Edition, 2001.

11. Peter Lichodzijewski, A. Nur Zincir-Heywood, and Malcom I.Heywood. *Dynamic Intrusion Detection Using Self-Organizing Maps*. In *The 14th Annual Canadian Information Technology Security Symposium (CITSS)*, 2002.

12. Peter Lichodzijewski, A. Nur Zincir-Heywood, and Malcom I.Heywood. *Host-based Intrusion Detection Using Self-Organizing Maps*. In *The IEEE World Congress on Computational Intelligence, International Joint Conference on Neural Networks, IJCNN'02*, 2002.

13. Hilmi Günes Kayac1k. *Hierarchical Self Organizing Map Based IDS on KDD Benchmark*. Master's thesis, Dalhousie University, 2003.
http://users.cs.dal.ca/~mheywood/Thesis/GKayacik.pdf

14. MIT Lincoln Laboratory. http://www.ll.mit.edu/IST/ideval/data/data index.html.

15. SOM_PAK. http://www.cis.hut.fi/research/som_lvq_pak.shtml.

16. Eskin E., Arnold A., Prerau M., Portnoy L., and Stolfo S. *A Geometric Framework for Unsupervised Anomaly Detection: Detecting intrusions in unlabeled data*. In D. Barbara and S. Jajodia, editors, Applications of Data Mining in Computer Security. Kluwer, 2002. ISBN 1-4020-7054-3, 2002.