Filtrando eventos de seguridad en forma conservativa mediante deep learning

Leandro Ferrado and Matías Cuenca-Acuna

Argentina Software Design Center (ASDC), Intel Security - Córdoba, Argentina
{leandro.ferrado,francisco.m.cuenca-acuna}@intel.com

Resumen Actualmente, los sistemas de seguridad utilizados en grandes organizaciones producen diariamente millones de eventos, que mediante otros productos se reducen a cientos de miles de alertas. Estas últimas deben ser analizadas por un conjunto reducido de personas, como los analistas de un Security Operations Center (SOC), lo cual supone un trabajo abrumador para realizar manualmente. Por mucho tiempo se desarrollaron soluciones modelando detectores de anomalías que luego cuesta trasladar a la industria efectivamente por requerir suficientes datos de ataques. En este trabajo se propone modelar un baseline que requiera principalmente datos de eventos "normales" para entrenarse, y que sirva para retener sólo aquellos que se desvían del comportamiento normal asimilado. Se utilizaron Stacked Auto-Encoders en el diseño, una técnica popular de deep learning en forma no supervisada, y los resultados obtenidos con datos de NSL-KDD impulsan la viabilidad de este enfoque para señalar incidentes con precisión sobre eventos de seguridad.

Keywords: eventos de seguridad, baseline, SOC, deep learning, stacked autoencoders

1. Introducción

En cualquier organización, las amenazas de seguridad se están volviendo cada vez más complejas, difíciles de detectar y pueden causar daños tanto en sus activos como en los procesos de negocios. Es por ello que no alcanza con valerse de un firewall, sistemas de detección de intrusos y productos anti-malware, por lo cual deben recurrir a un SOC. Un *Security Operations Center* o SOC es una unidad centralizada que trata con los problemas ocurridos en sistemas de información empresariales en un nivel técnico y organizacional. El mismo es responsable de monitorear, detectar, analizar, reportar y aconsejar sobre eventos e incidentes ocurridos en la organización, y para ello debe administrar sus productos de seguridad, dispositivos de red y de usuario final[1].

Se considera como un evento a cualquier ocurrencia observable en un sistema o red (e.g. archivos de log, emails, llamadas telefónicas, etc.), mientras que un incidente constituye un evento que actual o potencialmente pone en peligro la

[1] White Paper de McAfee® FoundStone®: http://www.mcafee.com/de/resources/white-papers/foundstone/wp-creating-maintaining-soc.pdf

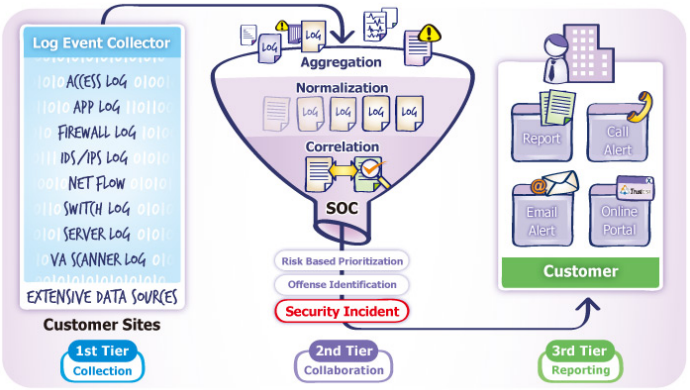


Figura 1: Niveles que componen el trabajo de un SOC en una organización.

confidencialidad, integridad, o disponibilidad de un sistema de información o conjunto de ellos . Los incidentes constituyen una amenaza de violación para las políticas de seguridad de una organización, y son típicamente detectados como desviaciones del estado normal de la red y sistemas. Analizando eventos, un analista de SOC debe ser capaz de separar eventos regulares (que constituyen una línea base de la actividad normal) de los eventos adversos o incidentes. A raíz de ello, existe actualmente un desafío de cerrar la brecha entre la cantidad de datos a manejar sobre eventos que es cada vez mayor y los limitados recursos humanos para identificar, analizar y actuar sobre los mismos [1].

En la Figura 1[2] se puede apreciar los niveles que estructuran el operar de un SOC, desde la colección de eventos provenientes de distintas fuentes, hasta el procesamiento de filtrado, agregación y correlación que retiene los incidentes, los cuales son comunicados a la organización cliente mediante alertas y reportes.

Para reducir la cantidad de datos a analizar, los SOC se valen de productos de seguridad que filtran los eventos que resultan importantes para revisar mediante algún criterio en particular. Los Sistemas de Detección de Intrusos (en inglés, *Intrusion Detection System* o IDS) tienen el propósito de alertar sobre anomalías y ataques que se encuentren en la red. Principalmente, se siguen dos enfoques:

a) *Basado en firmas*: mediante una base de datos con firmas de ataques, arrojando alertas cuando un evento coincide con alguna de ellas, que generalmente se corresponden con sistemas muy usados o aplicaciones cuyas vulnerabilidades de seguridad son conocidas.
b) *Basado en detección de anomalías*: generalmente se modela el comportamiento normal del recurso monitoreado, y luego se comparan nuevas entradas con el modelo entrenado para generar alertas sobre aquellas que se desvían significantemente en base a un criterio o métrica definida.

[2] En `http://www.citictel-cpc.com/EN/HK/Pages/product-services/security-operations-centres`

Mientras que los sistemas basados en firmas son generalmente favorecidos en productos comerciales debido a su previsibilidad y gran precisión, en investigaciones la detección de anomalías es frecuentemente concebida como una técnica más poderosa debido a su potencial teórico para descubrir nuevos patrones de incidentes que no son conocidos (o bien son variantes de los conocidos). En muchos trabajos intentan realizar dicha tarea con modelos clasificadores sobre datos de ataques (incluso reportando una elevada tasa de detección mayor al 98 % y con una tasa de falsas alarmas de 1 % [2]), pero que luego cuesta trasladar a herramientas comerciales por las limitaciones obvias de conseguir esa clase de información (que además se vuelve obsoleta rápidamente).

El objetivo de este trabajo es modelar un sistema para filtrar un flujo de eventos, cuyo entrenamiento se realice con eventos que se consideren regulares o normales, y que no necesariamente sea exhaustivo pero sí preciso en la detección de incidentes. Se considera que este tipo de sistemas constituye un valor importante para cualquier organización en términos de análisis y procesamiento, y con ello resulta viable implementar este enfoque para sistemas de seguridad que necesiten un filtrado de este tipo.

2. Materiales y métodos

Se procede a detallar la propuesta para conseguir el objetivo planteado en este trabajo. Para ello, primero se revisan conceptos básicos que justifican las técnicas elegidas y cómo se combinan para lograr el tipo de resultado buscado.

2.1. Datos utilizados

Para la experimentación, se optó por utilizar una base de datos pública conocida que ya se encuentre caracterizada para ahorrar el trabajo de extracción de características y pre-procesamiento. La mayor parte de las investigaciones hechas sobre detección de anomalías se realizan con los datos de KDDCup 99, que corresponden a flujos de red y cuyas características se clasifican en tres grupos.

1. **Básicas**: se presentan todos los atributos que pueden ser extraídos de una conexión TCP/IP.
2. **Contenido**: para valerse de información más específica de ataques, se utilizan algunas características que puedan revelar patrones propios de comportamiento sospechoso (e.g. número de intentos de login fallidos, número de archivos creados o accedidos, etc.).
3. **Tráfico**: se incluyen características que pueden ser computadas respecto a una ventana de intervalo, y se dividen a su vez en dos grupos:
   a) *De un mismo host*: se examinan sólo las conexiones de los últimos 2 segundos que tienen el mismo destino, y con ello se calculan estadísticas relacionadas al comportamiento de protocolo, servicios, etc.
   b) *De un mismo servicio*: se utilizan sólo las conexiones de los últimos 2 segundos que utilizan el mismo servicio.

Tabla 1: Porcentajes de aciertos obtenidos en [2] con distintos modelos en conjuntos de prueba sobre KDD y NSL-KDD.

	J48	**NBayes**	**NBTree**	**RF**	**RT**	**MLP**	**SVM**
KDDTest	**93.82 %**	81.66 %	93.51 %	92.79 %	92.53 %	92.26 %	65.01 %
KDDTest+	81.05 %	76.56 %	**82.02 %**	80.67 %	81.59 %	77.41 %	69.52 %
KDDTest-21	63.97 %	55.77 %	**66.16 %**	63.26 %	58.51 %	57.34 %	42.29 %

Tabla 2: Descripción básica de los conjuntos de datos utilizados.

	Normales	**Ataques**	**Total**
KDDTrain+20 % [3]	13449	11743	**25192**
KDDTest+	9710	12833	**22543**
KDDTest-21	2152	9698	**11850**

Esta base de datos ha sido criticada en la comunidad principalmente por tener mucha redundancia y también por su nivel de dificultad bajo ya que con simples modelos de machine learning se logra fácilmente una gran precisión sobre los datos de prueba (ver Tabla 1) y eso no asegura que se consiga el mismo desempeño en aplicaciones reales. Es por ello que en un trabajo se propuso mejorar esos aspectos de dicha base de datos [2], filtrando sus patrones a fin de obtener NSL-KKD, una nueva versión con 3 conjuntos mejores: KDDTrain+, KDDTest+, KDDTest-21. Para este trabajo se empleará el primero para entrenamiento y los otros dos para pruebas, con la particularidad de que no se utilizarán las etiquetas (es decir, la clasificación de los ataques) más que para distinguir en principio los datos "normales" de los "anómalos". Además, por simplicidad, se utilizarán sólo las características de *tráfico* de estos conjuntos ya que suponen información fácil de calcular y relevante para el tratamiento en cuestión. Por lo tanto, los datos utilizados comprenden un total de 19 características, y en la Tabla 2 se indica la cantidad de entradas que tienen los conjuntos por cada clase tratada.

2.2. Deep learning

El aprendizaje profundo (conocido en inglés como *deep learning*) es una rama de la inteligencia artificial que, debido al éxito de su utilización en problemas de gran complejidad, se ha vuelto popular y ampliamente desarrollado tanto a nivel empresarial como en el campo de la investigación. Actualmente esta técnica comprende el estado del arte en muchas aplicaciones (e.g. reconocimiento de rostros, procesamiento de la voz, etc.), especialmente con la implementación de autoasociadores como las Máquinas de Boltzmann Restringidas (RBM) y los Auto-Encoders (AE). Con estas técnicas se introduce el concepto de un esquema no supervisado como etapa inicial en el entrenamiento de redes neuronales [3].

Un Auto-Encoder es básicamente una red neuronal de tres capas a la cual, mediante un entrenamiento no supervisado con retropropagación, se obliga a que

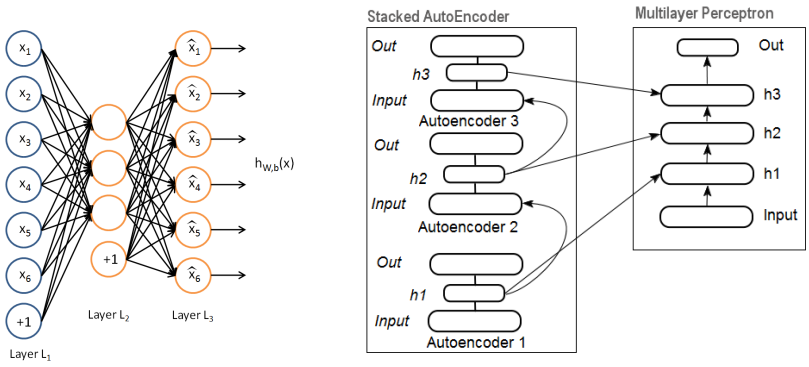
[3] Conjunto provisto por la fuente original de datos, originado con el 20 % de datos de KDDTrain+.

la salida de la red sea igual que la entrada (i.e. y = x). Con ello, se trata de que la capa oculta obtenga otra representación de la entrada (por lo general en distinta dimensión) y asegurando que la red neuronal aún así devuelve aproximadamente los mismos valores que en su entrada.

Para extraer abstracciones de alto nivel sobre los datos, los AEs son combinados en una red formando un Stacked Auto-Encoder (SAE). De esta forma, se busca capturar una representación interna de los datos en una red neuronal cuyos parámetros (pesos sinápticos y bias) se inicializan mejor que de forma estocástica. Entrenar redes profundas con esta técnica consta de dos etapas:

1. **Pre-entrenamiento**: El SAE se entrena de forma no supervisada secuencialmente, de forma que la salida de un AE es la entrada para entrenar el siguiente. Luego, la capa oculta de cada autoasociador se utiliza para inicializar las de una red neuronal estándar.
2. **Ajuste fino**: Una vez formada la red neuronal con sus parámetros inicializados, se sigue su entrenamiento supervisado convencionalmente. Se supone que con un buen pre-entrenamiento, el siguiente ajuste fino lleva poco tiempo para lograr valores óptimos en la salida.

En la Figura 2 se presenta la estructura de un AE y cómo se utiliza para construir SAEs[4]. Es preciso destacar que la capa de salida sólo se entrena en la etapa de ajuste fino, y la misma puede ser de clasificación o regresión dependiendo de la tarea asignada para la red. En este trabajo se utiliza una regresión de Softmax, la cual es muy utilizada en la salida de una red neuronal como capa de clasificación multi-clase, y para las dos clases tratadas (eventos normales e incidentes) equivale a realizar una regresión logística.



(a) Arquitectura de un AE (b) Etapas en la construcción de un SAE

Figura 2: Representación gráfica de un AE y el apilado de ellos para armar SAEs.

[4] Imagen a) extraida de: http://ufldl.stanford.edu/wiki/index.php/Autoencoders_and_Sparsity
Imagen b) extraida de: https://www.mql5.com/en/articles/1103

2.3. Desempeño buscado

En este trabajo se busca probar empíricamente que es posible modelar el *baseline* o comportamiento esperado de los eventos ocurridos en un sistema de seguridad, en particular valiéndose principalmente de datos considerados como "normales". Esto se considera factible por las siguientes razones:

1. Generalmente resulta difícil conseguir datos sobre eventos de seguridad debido a cuestiones confidenciales de las organizaciones.
2. Mucho más difícil que esos datos además incluyan ataques o incidentes.
3. En caso de poder conseguirse, es una tarea complicada etiquetar cada uno de ellos ya que la mayoría de los ataques comprenden una secuencia de patrones o dinámica temporal que no puede caracterizarse con ejemplos instantáneos.
4. A raíz de esto último, se puede considerar que incluso no tiene sentido diseñar un clasificador que sepa discriminar tipos específicos de los ataques detectados para una entrada dada.

Dado que parece casi utópico diseñar un modelo que sepa clasificar ataques o incidentes en un sistema de seguridad y que además no se vuelva rápidamente obsoleto, puede tener más sentido ofrecer un sistema que logre un filtrado con las siguientes funciones:

- Descartar lo mayor posible de eventos que suponen comportamiento normal.
- Retener eventos que efectivamente se desvían de lo asimilado como normal.

Las métricas establecidas para lograr dicho desempeño deseado son las de *precision* y *recall*. En un contexto de recuperación de información, el *precision* determina la cantidad de elementos relevantes del total que fueron recuperados, mientras que el *recall* representa la fracción de elementos recuperados del total que son relevantes[5]. Se puede decir que el esquema perseguido prioriza el *recall* sobre los datos de eventos normales que se utilizan para modelar y por ello se considera que el filtrado realizado es de forma "conservativa", pero a la vez "liberal" o con *precision* en lo que se quiere etiquetar como anómalo.

Así, se denota que el entrenamiento sigue un enfoque One-Class Classification (OCC) [4] ya que se ajusta el modelo con datos de una sola clase para identificarla sobre el resto posible, y para ello valiéndose de un esquema no supervisado que pueda caracterizar mejor el *baseline* sobre lo normal.

Existe un enfoque similar al seguido en este trabajo, que también utiliza las mismas técnicas para modelar el *baseline* de forma no supervisada [5]. En el mismo, se entrena un único AE profundo sólo con datos normales, que luego se utiliza para transformar todos los datos del conjunto de prueba y, mirando el error de reconstrucción cometido en la salida de la red, se establecen como anómalas aquellas entradas que producen un gran error (superando un umbral definido) ya que no conforman la estructura aprendida en el entrenamiento.

El enfoque de este trabajo trata de aprender la estructura con más de un nivel de abstracción posible mediante la apilación de AEs de forma consecutiva, y además se generaliza la detección de anomalías mediante una capa de clasificación al final de la red en lugar de fijar un umbral sobre el error de reconstrucción.

[5] Definición detallada en Wikipedia: `https://en.wikipedia.org/wiki/Precision_and_recall`

3. Resultados y conclusiones

Para el modelado se utilizó un framework de código abierto[6] diseñado por el autor de este trabajo, el cual permite entrenar redes neuronales con deep learning de forma distribuida mediante Apache Spark™ en Python. Se construyeron SAEs con dos configuraciones, una con sólo un nivel de abstracción y otra con dos. Los detalles de ello pueden verse en el Apéndice. Notar allí que se utilizaron técnicas de regularización que incluyen normas sobre los pesos sinápticos (L1 y L2) y un algoritmo de DropOut para el ajuste fino, los cuales son siempre recomendados en tutoriales de deep learning [6].

En base a las métricas de desempeño establecidas, se denota $P_{ataques}$ al valor de *precision* logrado sobre los datos que corresponden a ataques, mientras que R_{normal} refiere al *recall* obtenido para los datos que suponen un comportamiento normal. De forma complementaria, se tuvo en cuenta el *accuracy* en ataques $A_{ataques}$ y el porcentaje de reducción logrado $\%red$. A partir de ello, los resultados obtenidos en el modelado se detallan por cada etapa de construcción:

1. **Pre-entrenamiento**: Aquí se utilizaron únicamente datos de eventos normales, y se obtuvo en todas las pruebas que el modelo lograba $P_{ataques}$ y R_{normal} máximas (es decir, iguales a 1). No obstante, el $A_{ataques}$ variaba aproximadamente entre 0 y 0.05 en promedio, y esto confirma que se necesita un ajuste fino con datos de ambas clases para adiestrar a la capa clasificadora y que no se clasifique todo como normal.
2. **Ajuste fino**: En esta etapa se utilizan los mismos datos de entrenamiento, sumando una pequeña porción de ejemplos de la otra clase (e.g. un 10 % del total). Como sólo se busca mejorar el clasificador (y quizás también un poco las representaciones ocultas), este ajuste debe ser menos intensivo. Los resultados logrados se muestran en la Tabla 3 y Tabla 4. Notar allí que se obtuvo un ligero mejor desempeño con la red menos profunda (un único AE).

Tabla 3: Resultados con el modelo SAE-19-10-2 con un único AE.

	KDDTrain+20 %			**KDDTest+**			**KDDTest-21**			
	$P_{ataques}$	R_{normal}	$A_{ataques}$	$P_{ataques}$	R_{normal}	$A_{ataques}$	$P_{ataques}$	R_{normal}	$A_{ataques}$	% red
Media	0,9912	0,9987	0,6529	0,9908	0,9885	0,4091	0,9669	0,9688	0,2291	81,02
Desvío	0,0148	0,0003	0,0315	0,0021	0,0104	0,0084	0,0215	0,0177	0,0045	5,12

La comparación no puede ser hecha directamente sobre los resultados de la Tabla 1 debido a la diferencia de métricas utilizadas, pero en términos de desempeño se denota una mejoría importante entre lo obtenido respecto a lo esperado, incluso utilizando pocos datos de la clase correspondiente a anomalías (que podrían ser alteraciones artificiales de datos normales en lugar de ataques[7]).

[6] Repositorio en GitHub: `https://github.com/leferrad/learninspy`

[7] Una manera de implementar ello sobre datos de KDDCUP 99 se encuentra detallada en `https://www.mapr.com/ebooks/spark/08-unsupervised-anomaly-detection-apache-spark.html`

Tabla 4: Resultados con el modelo SAE-19-15-10-2 de dos AEs apilados.

	KDDTrain+			**KDDTest+**			**KDDTest-21**			
	$P_{ataques}$	R_{normal}	$A_{ataques}$	$P_{ataques}$	R_{normal}	$A_{ataques}$	$P_{ataques}$	R_{normal}	$A_{ataques}$	% red
Media	0,9901	0,9819	0,6113	0,9544	0,9703	0,3851	0,9121	0,8674	0,2241	76,42
Desvío	0,0139	0,0281	0,0808	0,0651	0,0444	0,0388	0,1209	0,1994	0,0261	7,62

Además, el mismo tratamiento se puede extrapolar prácticamente a cualquier otro sistema que intente detectar desviaciones sobre un flujo de datos.

A partir del buen desempeño obtenido, se puede concluir que tiene sentido implementar sistemas que modelen un *baseline* sobre datos que constituyan un comportamiento esperado, y que para ello se capture una representación interna de dichos datos utilizando técnicas avanzadas como las dadas por deep learning. En el caso tratado sobre seguridad, resulta factible de implementar en una organización que posiblemente no tenga muchos datos de ataques y necesite un sistema que alerte con la mayor precisión posible los incidentes que puedan surgir, para así iniciar análisis profundos sobre los eventos relacionados.

Referencias

1. Chisholm, C., & Khiabani, Hamed (2016). Boiling the Ocean: Security Operations and Log Analysis. SANS Institute Reading Room.
2. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009*.
3. Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning?. The Journal of Machine Learning Research, 11, 625-660.
4. Khan, S. S., & Madden, M. G. (2009). A survey of recent trends in one class classification. In Artificial Intelligence and Cognitive Science (pp. 188-197). Springer Berlin Heidelberg.
5. Arora, A., Candel, A., Lanford, J.,& Parmar, V. (2015). Deep Learning with H2O.
6. Li, F. F., & Karpathy, A. (2015). CS231n: Convolutional Neural Networks for Visual Recognition.

Apéndice: Configuraciones usadas para modelar

SAE-19-10-2:
Layer 0 with 19 neurons, using ReLU activation and 0.0 of DropOut ratio.
Layer 1 with 10 neurons, using ReLU activation and 0.5 of DropOut ratio.
Layer 2 with 2 neurons, using Softmax activation.
The loss is CrossEntropy for a task of classification.
L1 strength is 1e-06 and L2 strength is 5e-05.

SAE-19-15-10-2:
Layer 0 with 19 neurons, using ReLU activation and 0.0 of DropOut ratio.
Layer 1 with 15 neurons, using ReLU activation and 0.5 of DropOut ratio.
Layer 2 with 10 neurons, using ReLU activation and 0.5 of DropOut ratio.
Layer 3 with 2 neurons, using Softmax activation.
The loss is CrossEntropy for a task of classification.
L1 strength is 1e-06 and L2 strength is 5e-05.