ISBN 978–987–633–574–4

Detección de Patrones de Comportamiento en la Red a través del Análisis de Secuencias

Carlos Catania, Jorge Guerra, Juan Manuel Romero, Franco Palau, Gabriel Caffaratti, and Martín Marchetta

Universidad Nacional de Cuyo
Facultad de Ingeniería
Laboratorio de Sistemas Inteligentes (LABSIN)
Mendoza, Argentina
{harpo, jorge.guerra}@ingenieria.uncuyo.edu.ar
juaanromeero66@gmail.com
{franco.palau, gabriel.caffaratti,
martin.marchetta}@ingenieria.uncuyo.edu.ar

Resumen Los enfoques de detección por comportamiento en el tráfico de red se basan en encontrar patrones comunes que sigue un ataque a lo largo de su ciclo de vida, tratando de generalizarlos para poder detectar una traza de ataque no vista con anterioridad. Un enfoque común consiste en la generación de secuencias basadas en caracteres para representar comportamientos maliciosos, y luego aplicar modelos como Cadenas de Markov para generalizar a otros comportamientos similares. Sin embargo, estos últimos presentan limitaciones para explorar más allá del estado anterior. En el presente trabajo se analizan las ventajas y limitaciones de tres arquitecturas de redes neuronales para detectar comportamientos maliciosos capaces de recordar patrones vistos mucho tiempo atrás. Para esto se realizó una evaluación sobre un conjunto de datos específicamente diseñado que incluye comportamientos maliciosos y normales de diversas fuentes. Los resultados preliminares indican que, a pesar de su simplicidad, la aplicación de cualquiera de las arquitecturas de red es un enfoque válido para detectar comportamientos de red maliciosos, lo cual es prometedor para su aplicación a problemas de etiquetado de tráfico de red en el contexto de un flujo de trabajo con interacción humana.

Keywords: Botnet · Redes Neuronales · Seguridad Informática

1. Motivación

El presente trabajo se basa en la aplicación de modelos de comportamiento de trazas de red a partir del estudio de las características a largo plazo presentes en el tráfico. Los enfoques de detección basados en modelos de comportamiento han demostrado ser adecuados para hacer frente a los constantes cambios de actividades que presentan los ataques [2]. Un enfoque de detección por comportamiento se basa en encontrar aquellos patrones comunes que sigue un ataque a lo largo de su ciclo de vida, tratando de generalizarlos para ser capaz de detectar una traza de ataque no vista con anterioridad. Por ejemplo, en el caso de

las llamadas Botnets, periódicamente todos los bots necesitarán conectarse al Botmaster para recibir nuevas instrucciones. Esto constituye un claro patrón de comportamiento, que sólo es observado después de un largo período de tiempo.

En el marco del proyecto Stratosphere [5], se han implementado modelos de comportamiento en el sistema gratuito SLIPS (Stratosphere Linux Intrusion Prevention System [5]). SLIPS utiliza secuencias basadas en caracteres para representar comportamientos maliciosos, y luego aplica modelos como Cadenas de Markov (MCM, Markov Chain Models) para generalizar a otros comportamientos de este tipo. Los modelos basados en MCM se eligen por ser eficientes computacionalmente, pero no generalizan bien frente a ataques con nuevos comportamientos. Además los MCM tienen la limitación de que el cálculo de las probabilidades de transición depende únicamente del estado anterior.

Por otro lado, otras técnicas como ser las Redes LSTM (Long-Short Term Memory) [4], las Redes 1DCNN (1D Convolutional Neural Networks) [1], y más recientemente los modelos ATTE (Attention Models) [10], han demostrado ser eficientes para extraer patrones en secuencias y utilizarlos para clasificación [3]. Las arquitecturas LSTM y ATTE han sido exitosas en el tratamiento de patrones de dependencia a largo plazo, mientras que las 1DCNN han mostrado algunas limitaciones en estos casos. Sin embargo, la ventaja de las 1DCNN es que pueden entrenarse hasta 9 veces más rápido que las otras dos arquitecturas.

Al igual que [9], este trabajo se centra en analizar el problema de la detección del comportamiento Botnet utilizando las tres arquitecturas de redes neuronales mencionadas, con una complejidad mínima (con un número mínimo de capas). La hipótesis es que, a pesar de la simplicidad de las arquitecturas utilizadas, estas pueden aprender las propiedades comunes de los diferentes comportamientos de los ataques y el rendimiento resultante podría constituir una base para medir el de otras arquitecturas más complejas y costosas computacionalmente.

La principal contribución del trabajo es un análisis de las ventajas y limitaciones de tres arquitecturas de redes neuronales para detectar comportamientos maliciosos en la red. Para esto se realizó una evaluación sobre un conjunto de datos específicamente diseñado que incluye comportamientos maliciosos de 14 ataques reales diferentes y 6 comportamientos normales de diversas fuentes.

El trabajo se organiza como sigue. La Sección 2 presenta un modelo de representación de comportamiento basado en secuencias de caracteres. La Sección 3 describe las diferentes arquitecturas de las redes. La Sección 4 ofrece los detalles del diseño de los experimentos y sus resultados. La Sección 5 presenta las tareas de optimización de hiper-parámetros de los modelos. La Sección 6 muestra el desempeño de los modelos en ejemplos nuevos. Finalmente la Sección 7 expone las conclusiones del trabajo.

2. Representación del Comportamiento Mediante Secuencia de Caracteres

El enfoque de SLIPS modela el comportamiento de una traza de tráfico de red agregando los flujos según una 4-tupla compuesta por: la dirección IP de

origen, la dirección IP de destino, el puerto de destino y el protocolo. Todos los flujos de red que coinciden con una tupla se juntan y se los denomina *Conexión Stratosphere* (SC). A partir de una captura de tráfico se crean varias SC, cada una conteniendo un grupo de flujos. En base a esta representación la secuencia de comportamiento de una SC se calcula como sigue:

1. Se extraen tres características de cada flujo: tamaño, duración y periodicidad.
2. Se asigna a cada flujo un símbolo de *estado* según las características extraídas y la estrategia de asignación mostrada en la Tabla 1.
3. Después de la asignación, cada *conexión* tiene su propia cadena de símbolos que representa su comportamiento en la red.

Tabla 1. Estrategia de asignación de caracteres para la representación del comportamiento

	Tamaño Chico			**Medio**			**Grande**		
	Dur. Corta	**Dur. Med**	**Dur. Larga**	**Dur. Corta**	**Dur. Med**	**Dur. Larga**	**Dur. Corta**	**Dur. Med**	**Dur. Larga**
Per. Fuerte	a	b	c	d	e	f	g	h	i
Per. Débil	A	B	C	D	E	F	G	H	I
No-Per. Fuerte	r	s	t	u	v	w	x	y	z
No-Per. Débil	R	S	T	U	V	W	X	Y	Z
Sin Datos	1	2	3	4	5	6	7	8	9

	Diferencia de tiempo				
	0s a 5s	**5s a 1m**	**1m a 5m**	**5m a 1h**	**> 1h**
Símbolo	.	,	+	*	0

Un ejemplo de un *modelo de comportamiento basado en caracteres* se observa en la Figura 1, donde se muestran los símbolos que representan todo el flujo para una SC basada en el protocolo UDP.

2.4.R*R.R.R*a*b*a*a*b*b*a*R.R*R.R*a*a*b*a*a*a*a*

Figura 1. Un ejemplo de un SC desde la dirección 10.0.2.103, y destino a la dirección 8.8.8.8 utilizando el puerto 53 del protocolo UDP.

3. Arquitecturas de la Red Neural

Se consideraron tres arquitecturas de redes neuronales: (a) una basada en una red convolucional 1D (1DCNN) [1], (b) una red neuronal recurrente basada en LSTM [9], (c) una arquitectura que incluye un mecanismo de Atención [10]. El presente trabajo se enfoca en evaluar el rendimiento de las redes neuronales consideradas con un mínimo de capas adicionales además de las capas Convolucionales, LSTM y Atención. Una descripción visual de cada arquitectura se muestra en la Figura 2. A continuación se describen las capas utilizadas.

3.1. Capa Embedding

Esta capa permite proyectar secuencias de caracteres de entrada de longitud l en una secuencia de vectores R^{lxd}, donde l debe determinarse a partir de la

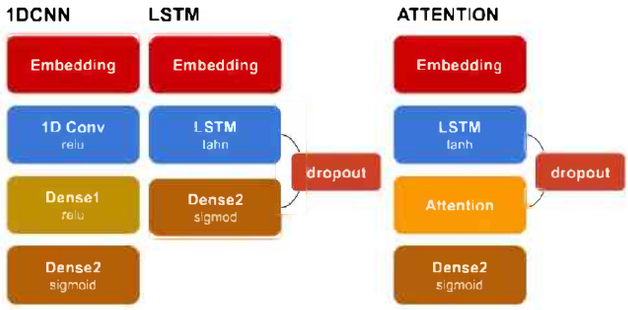

Figura 2. Esquema simplificado de las arquitecturas de las tres redes aplicadas al problema de clasificación de secuencias.

información proporcionada por las secuencias del conjunto de entrenamiento, y d es un parámetro libre que indica la dimensión de la matriz resultante [9]. Mediante esta capa la red neuronal aprende de manera eficiente el conjunto óptimo de características que representan los datos de entrada.

3.2. Capa 1D Convolucional

Para el problema de detección de comportamiento se utiliza una capa convolucional de una dimensión, compuesta por un conjunto de filtros convolucionales que se aplican a diferentes porciones de la secuencia.

Al aplicar el mismo filtro en toda la secuencia se reduce el tiempo de cálculo en comparación con las arquitecturas tradicionales como el Perceptrón Multicapa (MLP). Además, dado que un núcleo convolucional opera de forma independiente sobre cada 4-grama, es posible recorrer toda la capa de entrada de forma simultánea. Estas características aportan ventajas sobre otros enfoques de aprendizaje profundo que se suelen utilizar para el procesamiento de textos, como ser la LSTM [4, 8, 9].

3.3. Capa LSTM (Long Short-Term Memory)

La capa LSTM [4] ha sido aplicada con éxito a problemas de Procesamiento de Lenguaje Natural (NLP). La principal ventaja de las redes que usan una capa LSTM es que son capaces de memorizar información vista previamente en una secuencia. Esta capacidad se traduce en el aprendizaje de patrones de caracteres que no son necesariamente contiguos en la secuencia, sino que pueden haberse observado temporalmente muy distanciados.

3.4. Capa de Atención

A pesar de su capacidad para recordar patrones vistos muy atrás en el tiempo, las redes LSTM tienen problemas para detectar patrones cuando la secuencia supera un determinado tamaño. Para atacar este problema surge la idea de

construir un vector de contexto que preste especial "atención" a determinados estados ocultos de la capa LSTM (en lugar de sólo el último estado) [10]. Luego el modelo mismo aprende a cuáles elementos de la secuencia prestar atención y a cuáles es mejor ignorar. Los modelos de atención han probado ser una mejora significativa respecto a las redes LSTM.

3.5. Capa Densa

Esta es una capa clásica de una red de tipo MLP totalmente conectada. Esta capa se aplica sobre las características extraídas por la arquitectura 1DCNN, y como parte de las tres arquitecturas consideradas para obtener la probabilidad de que una secuencia pertenezca a la clase *Botnet* o *Normal*.

4. Diseño Experimental

4.1. Métricas

Para evaluar las arquitecturas propuestas, se utilizan las siguientes métricas estándares: **Sensibilidad**, **Especificidad** y **Exactitud Balanceada**. La Sensibilidad se calcula como el cociente entre los elementos correctamente identificados como positivos sobre el total de verdaderos positivos. La Especificidad se calcula como el cociente entre los elementos de la clase negativa identificados como negativos sobre el número total de ejemplos de la clase negativa. La Exactitud Balanceada es la media entre la Sensibilidad de ambas clases, donde la clase positiva contiene las SC que posee comportamiento malicioso.

4.2. Descripción del conjunto de datos

Las tres arquitecturas de redes neuronales fueron entrenadas y evaluadas con un conjunto de datos de seguridad de redes específicamente diseñado para el problema de la detección de botnets, conformado por veinte capturas de red publicadas por el grupo de investigación Stratosphere IPS de la Czech Technical University (CTU) [6]. El conjunto de datos referido como **CTU [A]** contiene cinco capturas de botnet y cuatro capturas normales, las cuales incluyen tráfico como DNS, HTTPS y P2P. En total, todas las capturas representan 20747 SC, teniendo 17826 etiquetadas como "Botnet" y 1449 etiquetadas como "Normal". Todas estas capturas fueron recogidas entre 2013 y 2017.

Además, se utilizó un segundo conjunto de datos, denominado **CTU [B]** para realizar una evaluación final e independiente de los modelos resultantes. El CTU [B] se compone de cinco redes de bots y dos capturas normales. En total, las siete capturas representan 1574 SC con 1437 etiquetadas como "Botnet" y 137 etiquetadas como "Normal".

La Tabla 2 proporciona una breve descripción de cada captura para ambos conjuntos de datos. La primera columna determina el conjunto de datos. Luego, las dos siguientes columnas muestran el ID de la captura según la denominación

utilizada por en el proyecto Stratoshpere y el tipo de tráfico incluido en la captura (es decir, Botnet o Normal). A continuación, en las columnas cuatro y cinco, el nombre del malware y el número de SC incluidas en la captura. En muchos casos el nombre del malware no se encontraba disponible, por lo tanto, se asume que todas las capturas marcadas como desconocidas no difieren significativamente en el tipo de ataque utilizado.

Tabla 2. Capturas de tráfico en los conjuntos CTU [A] y CTU [B]

Conj.	MCFP Captura ID	Clase	Malware	N. Con.
CTU [A]	2013-10-01_capture-win8	botnet	desconocido	3463
	2013-10-01_capture-win12	botnet	desconocido	2197
	2013-08-20_capture-win15	botnet	Kelihos	9844
	2013-11-06_capture-win6	botnet	Zbot	2088
	2014-03-12_capture-win3	botnet	Zbot	234
	2017-04-30_win-normal	normal	-	348
	2017-05-02_kali-normal	normal	-	562
	2017-04-18_win-normal	normal	-	387
	2017-04-19_win-normal	normal	-	152
CTU [B]	2013-08-20_capture-win2	botnet	Zbot	28
	2014-01-25_capture_win3	botnet	Zbot	52
	2014-02-10_capture-win3	botnet	Zbot	47
	2013-11-25_capture-win7-2	botnet	desconocido	627
	2014-01-31_capture-win7	botnet	desconocido	581
	2014-01-14_capture-win2	botnet	Hicrazyk.A	102
	2013-12-17_capture1	normal	-	32
	2017-04-25_win-normal	normal	-	105

5. Ajuste de Hiper-parámetros

Para ajustar los parámetros de las arquitecturas presentadas en la Sección 3, se utilizó un barrido matricial. La gama completa de valores de los hiper-parámetros se presenta en la Tabla 3. Las dos primeras columnas se refieren a los parámetros y a los valores considerados durante la búsqueda de parámetros, mientras que las columnas restantes presentan los valores óptimos obtenidos.

Tabla 3. Rango de los hiper-parámetros y los valores óptimos encontrados para las tres arquitecturas. Los dos modelos con menor variabilidad fueron seleccionados.

Hiper-parámetros	Valores	1DCNN		LSTM		ATTE	
		(1)	(2)	(1)	(2)	(1)	(2)
Tamaño LSTM	128,64,32	-	-	64	64	128	64
Filtros 1DCNN	256,128,64	256	128	-	-	-	-
Kernel 1DCNN	8,4,2	4	4	-	-	-	-
Tamaño Embedding	128,64,32	128	128	32	64	128	32
Tamaño Denso	256,128,64	128	256	-	-	-	-
Valor Dropout	0.5,0.1	-	-	0.1	0.1	0.5	0.1
Tamaño Batch	1024,256	1024	1024	1024	256	1024	256

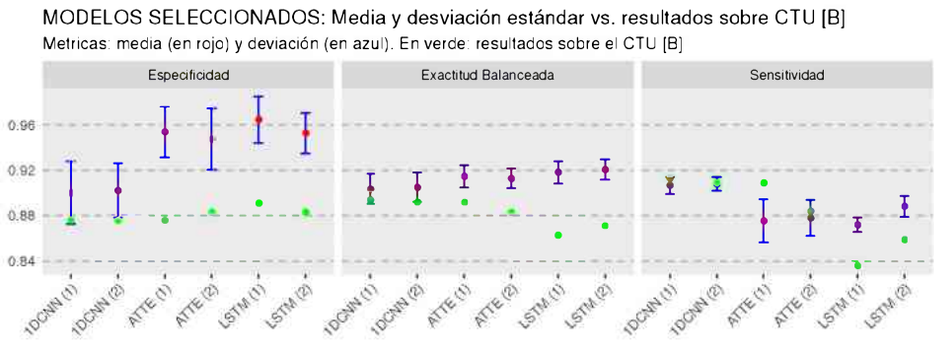

Figura 3. Media y desviación estándar para cada uno de los 6 modelos que obtuvieron mejores resultados.

A fin de evaluar cada conjunto de hiper-parámetros en un plazo de tiempo aceptable se analizó el porcentaje mínimo de muestras necesarias para el entrenamiento/prueba sin que afecte significativamente al rendimiento del modelo. Se muestrearon 30 pares de conjuntos de datos sobre un subconjunto del 25 % de CTU [A], donde el 20 % de las muestras se seleccionaron para entrenamiento y el 5 % restante se utilizó para la prueba.

No se observaron diferencias significativas en las métricas de rendimiento durante la búsqueda matricial. Por esto, se seleccionaron los modelos con un valor medio de Exactitud Balanceada superior a 0,89, y la menor varianza.

La Figura 3 complementa los resultados de la Tabla 3 con los valores de la media y la desviación estándar para las tres métricas consideradas. En general, todas las arquitecturas mostraron una baja variabilidad, en particular para las métricas de Exactitud Balanceada y Sensibilidad (número de botnets correctamente detectados). Sin embargo, en la Especificidad (número de conexiones normales correctamente detectadas), se observa una variabilidad considerablemente mayor. Esto último podría explicarse por el bajo número de ejemplos normales presentes en el conjunto de datos y su alta variabilidad.

Por otro lado, las arquitecturas basadas en 1DCNN mostraron los mejores resultados en cuanto a la detección correcta de los comportamientos de las botnets (Sensibilidad) mientras que ATTE y LSTM observaron un mejor rendimiento cuando se trataba de comportamientos normales (Especificidad). Sin embargo, al considerar la métrica de Exactitud Balanceada, las tres arquitecturas de red no parecieron mostrar diferencias significativas.

6. Evaluación sobre nuevos ejemplos

Se realizó una evaluación independiente de los modelos con los hiper-parámetros seleccionados en la Sección 5. Esta vez, las tres arquitecturas de red se entrenaron con CTU [A] completo y se evaluaron contra CTU [B] completo. Los resultados se muestran en la Tabla 4.

En general, todas las arquitecturas de red mostraron valores aceptables en todas las métricas consideradas (es decir, presentaron valores superiores al 80 %). Al igual que los resultados observados en la Sección 5, la 1DCNN mostró el mejor valor de Sensibilidad mientras que las redes LSTM funcionan mejor en términos de Especificidad. Sin embargo, cuando se consideran los valores para la Exactitud Balanceada, la red 1DCNN supera claramente a las redes LSTM. Por otro lado, las redes con una capa de atención mostraron un rendimiento similar al de 1DCNN en todas las métricas consideradas.

A pesar de los resultados similares entre las tres arquitecturas de redes neuronales (ver Tabla 4), surgen algunas diferencias cuando analizamos capturas específicas de comportamientos normales y de botnets. En la Figura 4, se observan los valores de Sensibilidad para cada una de las capturas utilizando los mejores modelos en cada arquitectura considerada. Con la excepción de la captura de `2014-01-14-win2`, todas las capturas de botnets fueron detectadas con una Sensibilidad superior al 80 % por cada una de las arquitecturas de red.

Sin embargo, las redes 1DCNN y ATTE ofrecen un mejor rendimiento en comparación con LSTM para todas las capturas de botnets. Como se muestra en el boxplot de la parte inferior de la figura, 1DCNN y ATTE mostraron valores de Sensibilidad superiores al 90 % para las capturas de botnets. En particular, la captura de botnet `2013-11-25-capture-win7-2` con 625 SC es la responsable del bajo rendimiento de Sensibilidad de LSTM. Según se puede observar, varias SC no son detectadas por las arquitecturas LSTM (79 % de Sensibilidad), mientras que 1DCNN y ATTE muestran un rendimiento de detección considerablemente mejor de 91 % y 89 %, respectivamente.

En cuanto a las capturas normales, no se observan diferencias significativas entre todas las arquitecturas de red. En particular, todos los modelos mostraron un bajo rendimiento de detección para la captura `2013-12-17-capture1`. Sin embargo, casi todas las SC de la captura normal `2017-04-25-win-normal` (105 SC en total) son detectadas correctamente por LSTM (es decir, 99 % de Sensibilidad), mientras que para los modelos 1DCNN y ATTE la Sensibilidad disminuye al 97 %. Esto último puede explicar el mejor rendimiento mostrado por LSTM en la métrica de Especificidad de la Tabla 4.

Por otro lado, la captura de botnet `2014-01-14-capture-win2` es la única que podemos corroborar que fue diferente a los comportamientos de botnet

Tabla 4. Modelos seleccionados entrenados con la totalidad del CTU19[A] y evaluados sobre CTU19[B].

Arquitectura de la red	Especificidad	Sensibilidad	Exactitud Balanceada
ATTE (1)	0.876	0.909	0.892
ATTE (2)	0.883	0.884	0.883
LSTM (1)	0.891	0.836	0.863
LSTM (2)	0.883	0.859	0.871
1DCNN (1)	0.876	0.913	0.894
1DCNN (2)	0.876	0.909	0.892

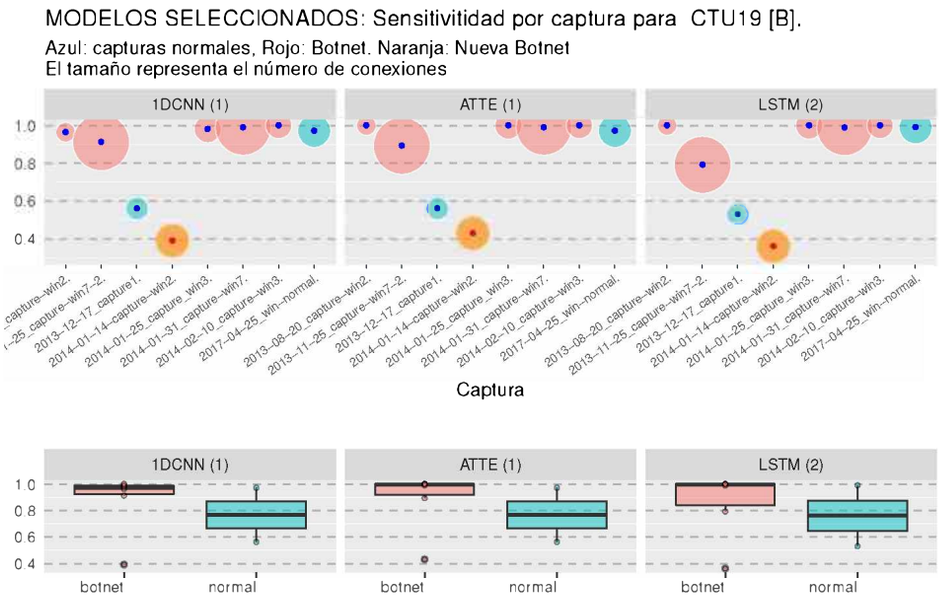

Figura 4. Arriba: Valores de Sensibilidad discriminados por captura de trafico. **Abajo:** Sensibilidad discriminada por clase.

utilizados durante el entrenamiento. En cuanto a la Sensibilidad, todas las arquitecturas de red detectaron menos del 50 % de las SC presentes. Para esta captura en particular, el mejor rendimiento de detección lo mostraron los modelos ATTE con un 43 % mientras que 1DCNN y LSTM mostraron valores de sensibilidad por debajo del 40 %. Frente a esta disminución, parece claro que el comportamiento de esta nueva Botnet difiere del resto de las Botnets. Sin embargo resulta interesante observar que todos los modelos lograron cierto nivel de generalización.

7. Conclusiones y trabajos futuros

Se han evaluado tres arquitecturas de redes neuronales sencillas y bien establecidas para la clasificación de secuencias en un conjunto de datos compuesto por 20 capturas de tráfico de diferentes comportamientos maliciosos y normales. En general parece que, a pesar de su simplicidad, la aplicación de cualquiera de las arquitecturas de red es un enfoque válido para detectar comportamientos de red malicfiosos. Todos los modelos han mostrado una precisión equilibrada por encima del 80 %. A pesar de una evidente reducción del rendimiento, todos los modelos también han mostrado un rendimiento aceptable cuando se introdujeron nuevos comportamientos de botnet en la evaluación independiente (el 40 % de las SC se detectaron correctamente).

Sorprendentemente la arquitectura 1DCNN, a pesar de no ser capaz de capturar dependencias a largo plazo, ha mostrado resultados aceptables a lo largo

de todo el proceso de evaluación (mostró un valor de Exactitud balanceada entre el 88 % y el 92 %). Este buen rendimiento en términos de detección de ataques y de trazas normales hacen de las redes basadas en 1DCNN una buena opción para detectar los comportamientos de la red. Esto cobra más importancia si se tiene en cuenta que el re-entrenamiento periódico requerido por el desvío de la distribución de datos se ha convertido en un proceso común en aplicaciones de aprendizaje automático, y que 1DCNN requiere un tiempo considerablemente menor durante el proceso de entrenamiento.

A futuro se planea la evaluación de estas arquitecturas y otras más complejas a problemas de etiquetado de trafico de red en el contexto de un flujo de trabajo con interacción humana [7].

8. Agradecimientos

Los autores agradecen el apoyo recibido por la SIIP-UNCuyo y la ANPCyT (proyectos 06/B363, 06/B374 y PICT 1435), y el de NVIDIA Corporation (donación de una GPU Titan V).

Referencias

1. Catania, C., Garcia, S., Torres, P.: An analysis of convolutional neural networks for detecting DGA. In: Proceedings of XXIV Congreso Argentino de Ciencias de la Computación (2018)
2. Garcia, S.: Identifying, Modeling and Detecting Botnet Behaviors in the Network. Ph.D. thesis, UNICEN University (2014). https://doi.org/10.13140/2.1.3488.8006
3. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. Adaptive computation and machine learning, The MIT Press, Cambridge, Massachusetts (2016)
4. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation **9**(8), 1735–1780 (11 1997)
5. Stratosphere Research Laboratory: Stratosphere IPS for Linux (2019), https://www.stratosphereips.org/stratosphere-ips-for-linux, [Online; accessed May-2021]
6. Stratosphere Research Laboratory: Malware Capture Facility Project (2020), https://www.stratosphereips.org/datasets-malware, [Online; accessed May-2021]
7. Torres, J.L.G., Catania, C.A., Veas, E.: Active learning approach to label network traffic datasets. Journal of Information Security and Applications **49**, 102388 (Dec 2019)
8. Torres, P., Catania, C., Garcia, S., Garino, C.G.: An analysis of Recurrent Neural Networks for Botnet detection behavior. In: 2016 IEEE Biennial Congress of Argentina (ARGENCON). pp. 1–6. IEEE, Buenos Aires, Argentina (Jun 2016)
9. Woodbridge, J., Anderson, H., Ahuja, A., Grant, D.: Predicting domain generation algorithms with long short-term memory networks. ArXiv (2016), http://arxiv.org/abs/1611.00791
10. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical Attention Networks for Document Classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 1480–1489. Association for Computational Linguistics, San Diego, California (2016)