SOLICITUD DE TEMA DE TRABAJO FINAL

CARRERA: Ingeniería en Informática

DATOS ALUMNO			
APELLIDO Y NOMBRE Torres, Pablo Daniel			DOCUMENTO 32571900
DOMICILIO Sargento Cabral 471 Dpto C			LOCALIDAD Mendoza Capital
PROVINCIA Mendoza	C.P. 5500	TELEFONO 2616145073	E-mail pablo.dtorres@gmail.com

DATOS TRABAJO FINAL
TÍTULO DEL TRABAJO FINAL Redes neuronales recurrentes para la detección de comportamiento de Botnets
ASESOR Carlos Catania
CO ASESOR Diego Navarro, Sebastián García
ÁREA CONOCIMIENTO Machine Learning, Seguridad Informática, TCP/IP
MATERIAS RELACIONADAS Teleinformática, Inteligencia Artificial

Mendoza, de de 20 .-

_____ _____

Alumno Asesor

DESCRIPCIÓN DEL ANTEPROYECTO

(Adaptar el tamaño del documento según sea necesario. Consultar Art. IV y X del Reglamento de Trabajo Final)

TITULO

Redes neuronales recurrentes para la detección de comportamiento de Botnets

INTRODUCCIÓN

Tradicionalmente, las técnicas de detección de aplicaciones maliciosas (también denominadas **malware**) estaban basadas en dos enfoques. Por un lado los sistemas de reconocimiento basados en reglas que describen características (**firmas**) de la carga (o **payload**) del tráfico malicioso y por otro lado los sistemas basados en modelos estadísticos para detección de anomalías. Sin embargo con el incremento en la complejidad de las acciones provocadas por aplicaciones maliciosas, la detección se fue orientando a **sistemas basados en detección por comportamiento** que sean capaces de capturar los cambios en el comportamiento de la aplicación a lo largo del tiempo.

La hipótesis detrás de estos sistemas es que existe un conjunto de operaciones inherentes a cada aplicación/software que opera en la red. Este conjunto de operaciones es transformado en una secuencia de datos transmitidos por la red. Luego, a partir de la observación y análisis de estas secuencias de datos es posible reconocer un comportamiento propio de la aplicación reconocible a lo largo del tiempo. Este tipo de enfoque ha demostrado poseer ventajas significativas: (i) de manera similar a los enfoques basados en firmas, estos pueden ser fácilmente desplegados y compartidos en otras infraestructuras de red, (ii) al igual que los sistemas de detección por anomalías, estos pueden detectar comportamiento malicioso no visto con anterioridad.

El proceso para la construcción de un modelo de comportamiento asociado a un malware consiste de tres etapas. En una primera etapa se genera un modelo de comportamiento considerando diferentes características de la red (por ej. tamaño de paquete, periodicidad, duración, etc.). Este modelo permite visualizar de manera sencilla un gran número de conexiones de red a lo largo del tiempo y constituye además una herramienta de análisis. Luego, en una segunda etapa de reconocimiento se realiza un proceso de asociación entre el modelo de comportamiento y la aplicación maliciosa, esta tarea normalmente denominada **etiquetado**, requiere del apoyo de un especialista humano. Finalmente, en una tercera etapa, mediante el uso de **algoritmos con base estadística** o de **aprendizaje automático** se generaliza el modelo de comportamiento a fin de mejorar la capacidad del modelo para reconocer de forma automática otras aplicaciones maliciosas que presenten un comportamiento similar.

Bajo este contexto, uno de los algoritmos que surge como mejor alternativa para abordar el problema son las **redes neuronales recurrentes** (Recurrent neural networks o RNN). El potencial de estos algoritmos radica en su capacidad para analizar conjuntos de secuencias de tamaño variable, usualmente haciendo referencia a secuencias temporales. La red analiza la secuencia un estado a la vez, manteniendo una *memoria* que le permite contextualizar al estado dado que es consciente de los estados anteriores. Dentro de los tipos de RNN, uno de los más conocidos y probados tipos de redes es el modelo Long Short Term Memory (LSTM), que en la práctica se ha probado superior al algoritmo original, sobre todo por su capacidad para mantener las dependencias entre los estados con secuencias de tamaño considerable, lo que las convierte en uno de los mejores candidatos para abordar el problema.

OBJETIVOS

Analizar el potencial de las **redes neuronales recurrentes** (RNN) para el reconocimiento y la detección de comportamiento malicioso en el tráfico de red.

Objetivos Secundarios

1. Realizar una revisión de literatura en la aplicación de RNN a detección de comportamiento malicioso en el tráfico de red.
2. Implementar una variante de RNN para realizar el proceso de detección de comportamiento malicioso.
3. Evaluar la performance de la RNN en términos de detección de comportamiento malicioso y su capacidad de generalización.

DESCRIPCIÓN TÉCNICA

La herramienta principal sobre la que se va a basar este trabajo es **Stratosphere Testing Framework (STF)**. El concepto detrás de STF es ofrecer una plataforma sobre la cual se puedan manejar, verificar y evaluar distintos tipos de algoritmos abocados a la detección de malware. STF es capaz de cargar distintos tipos de datasets (pcap, netflow, flow, etc) y generar los modelos de comportamiento que surgen de los mismos. Los modelos de comportamiento están compuestos por una secuencia de estados que son el resultado del análisis de la periodicidad, tamaño y duración del flujo de datos bidireccional entre el host y el servidor, encuadrados bajo los parámetros de STF que definen el estado que le corresponde dependiendo del valor que toman dichos atributos a lo largo de la vida del flujo. STF además permite el etiquetado de cada modelo, por tanto manteniendo en la base de datos el tipo de tráfico que identifica a cada modelo específico. Los modelos de comportamiento con su etiqueta serán lo que se utilizará como entrada para el

entrenamiento y evaluación del algoritmo de RNN que se implementará en este trabajo .

Para llevar a cabo la implementación del algoritmo se propone utilizar Keras, un framework modular de redes neuronales desarrollado en Python que funciona por encima del framework Theano, el cual permite manejar de manera eficiente expresiones matemáticas con arreglos multidimensionales. El razonamiento detrás de esta decisión radica en la facilidad que presenta el framework para prototipar nuevas ideas y materializarlas en resultados en el menor tiempo posible.

Dentro de las variantes de RNN se decanta por LSTM (Long Short Term Memory), de probada eficacia para el análisis de secuencias, como las utilizadas para generar los modelos de STF

Para disminuir los tiempos de entrenamiento se utilizará aceleración por GPU a través de la plataforma de NVIDIA CUDA, cuya integración en Theano hace que su implementación sea transparente a la programación del algoritmo.

FASES DE TRABAJO Y ESTIMACIÓN TEMPORAL

La metodología de trabajo puede dividirse en 3 etapas.

1. . Introducción
    1. Una primera etapa para tomar conocimiento de los problemas asociados a la detección de comportamiento malicioso como así también los algoritmos actualmente implementados. En esta etapa se realizará un acercamiento al **Strastosphere Testing Framework** (STF). Esta etapa no solo incluye tomar conocimiento de los procesos básicos para su instalación, configuración y

utilización general sino también tomar conocimiento del funcionamiento interno de los algoritmos implementados hasta el momento en STF. Para esto último, se tomará como base los artículos y la tesis de doctorado del Dr. Sebastián García.

2. En una segunda etapa se realizará una breve revisión de la literatura sobre la aplicación de RNN a problemas de detección de comportamiento malicioso. La idea detrás de esta etapa es tratar de determinar cuáles han sido los beneficios e inconvenientes al aplicar este tipo de algoritmos.

1. Desarrollo de un algoritmo para el reconocimiento de comportamiento malicioso basado en RNN.
    1. Se desarrollará un primer prototipo basado en RNN. Este primer prototipo será desarrollado en python utilizando alguna de las bibliotecas disponibles que permitan la implementación de RNN de manera simple y eficiente. En esta etapa se definirán los diferentes aspectos de la red como ser: el tipo de RNN a utilizar, la topología de la red y la secuencia de entrada entre otras.

1. Evaluación Experimental
    1. Se determinará la capacidad del algoritmo para discriminar el tráfico de botnet del tráfico normal. Durante este proceso se consideran las métricas habituales en el área utilizando mecanismos para garantizar la generalidad del modelo obtenido como validación cruzada.

    2. Se evaluarán diferentes aspectos del algoritmo propuesto como ser los diferentes tamaños en los conjuntos de entrenamientos, el tamaño de secuencia y la capacidad de lidiar con conjuntos de datos no balanceados. Este último experimento apunta a evaluar

la robustez del algoritmo para una futura implementación en escenarios de tráfico real.

3. A fin de evaluar su capacidad de generalización, el algoritmo desarrollado se evaluará considerando diferentes conjuntos de datos provenientes de diferentes tipos de botnets no utilizadas en el entrenamiento del algoritmo.

Estimación Temporal

Actividades	Septiembre				Octubre				Noviembre				Diciembre				Enero			
	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4
Acercamiento																				
Desarrollo del algoritmo																				
Experimentación																				

BIBLIOGRAFÍA

[1] Y. Bengio, R. Ducharme, and P. Vincent, “A Neural Probabilistic Language Model.”

[2] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, J. Kandola, T. Hofmann, T.

Poggio, and J. Shawe-Taylor, “A Neural Probabilistic Language Model,” J. Mach. Learn. Res., vol. 3, pp. 1137–1155, 2003.

[3] G. C. Cawley and N. L. C. Talbot, “On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation,” J. Mach. Learn. Res., vol. 11, p. 2079−2107, 2010.

[4] N. V Chawla, “DATA MINING FOR IMBALANCED DATASETS : AN OVERVIEW.”

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Comparison of balancing techniques for unbalanced datasets,” J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.

[6] T. Dietterich, “Machine learning for sequential data: A review,” Struct. syntactic, Stat. pattern Recognit., pp. 1–15, 2002.

[7] T. G. Dietterich, “Approximate Statistical Tests for Comparing. Supervised Classification Learning Algorithms.,” Neural Comput., vol. 10, no. 7, pp. 1895 – 1923, 1998.

[8] P. Domingos, “A few useful things to know about machine learning,” Commun. ACM, vol. 55, no. 10, p. 78, 2012.

[9] V. Ganganwar, “An overview of classification algorithms for imbalanced datasets,” Int. J. Emerg. Technol. Adv. Eng, vol. 2, no. 4, pp. 42–47, 2012.

[10] S. García, “Identifying , Modeling and Detecting Botnet Behaviors in the Network Universidad Nacional del Centro de la Provincia de Buenos Aires Doctoral Thesis Identifying , Modeling and Detecting Botnet Behaviors in the Network,” no. August, 2015.

[11] S. García, M. Grill, J. Stiborek, and A. Zunino, “An Empirical Comparison of Botnet Detection Methods,” Comput. Secur., vol. 45, no. 0, pp. 100 – 123, 2014.

[12] S. García, V. Uhlíř, and M. Rehak, “Identifying and modeling botnet C&C behaviors,” Proc. 1st Int. Work. Agents CyberSecurity - ACySE ’14, no. MAY 2014, pp. 1–8, 2014.

[13] A. Graves, “Supervised Sequence Labelling with Recurrent Neural Networks.,” Lancet, vol. 346, no. 8988, p. 1501, 1995.

[14] G. Gu, J. Zhang, and W. Lee, “BotSniffer : Detecting Botnet Command and Control Channels in Network Traffic,” in Proceedings of the 15th Annual Network and Distributed System Security Symposium., 2008, vol. 53, no. 1, pp. 1–13.

[15] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” arXiv: 1207.0580, pp. 1–18, 2012.

[16] R. Jozefowicz, “An Empirical Exploration of Recurrent Network Architectures,” Proc. …, vol. 37, 2015.

[17] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, “Handling imbalanced datasets : A review,” GESTS Int. Trans. Comput. Sci. Eng., vol. 30, no. 1, pp. 25–36, 2006.

[18] Z. C. Lipton, “A Critical Review of Recurrent Neural Networks for Sequence Learning,” 2015.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space.”

[20] A. Y. Ng, “Preventing Overfitting of Cross-Validation Data,” 2010.

[21] T. M. Padmaja, N. Dhulipalla, R. S. Bapi, and P. R. Krishna, “Unbalanced data classification using extreme outlier elimination and sampling techniques for fraud detection,” 15th Int. Conf. Adv. Comput. Commun. (ADCOM 2007), pp. 511–516, Dec. 2007.

[22] R. B. Rao and G. Fung, “On the Dangers of Cross-Validation . An Experimental Evaluation,” Solutions, pp. 588–596, 2006.

[23] C. Rossow, C. J. Dietrich, C. Grier, C. Kreibich, V. Paxson, N. Pohlmann, H. Bos, and M. Van Steen, “Prudent Practices for Designing Malware Experiments: Status Quo and Outlook,” in 2012 IEEE Symposium on Security and Privacy, 2012, pp. 65–79.

[24] R. C. Staudemeyer, “The importance of time: Modelling network intrusions with long short-term memory recurrent neural networks,” no. November, p. 224, 2011.

[25] Z. Xing, J. Pei, and E. Keogh, “A brief survey on sequence classification,” ACM SIGKDD Explor. Newsl., vol. 12, no. 1, p. 40, 2010.

[26] “Addressing the curse of imbalanced training sets: one-sided selection,” vol. 4.

[27] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley and Y. Bengio. "Theano: new features and speed improvements". NIPS 2012 deep learning workshop.

[28] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley and Y. Bengio. "Theano: A CPU and GPU Math Expression Compiler". Proceedings of the Python for Scientific Computing Conference (SciPy) 2010. June 30 - July 3, Austin, TX

OBSERVACIONES