An Analysis of Recurrent Neural Networks for Botnet Detection Behavior

Pablo Torres*[1], Carlos Catania †[2], Sebastian Garcia‡[3] and Carlos Garcia Garino§[4]

*FING, Universidad de Mendoza Mendoza, Argentina
[1]pablo.dtorres@gmail.com
†FCEyN - ITIC, Universidad Nacional de Cuyo Mendoza, Argentina
[2]ccatania@itu.uncu.edu.ar
‡CTU - Czech Technical University Prague,Czech Republic
[3]sebastian.garcia@agents.cvut.fel.cz
§FING - ITIC, Universidad Nacional de Cuyo Mendoza, Argentina
[4]cgarcia@itu.uncu.edu.ar

***Abstract*—A Botnet can be conceived as a group of compromised computers which can be controlled remotely to execute coordinated attacks or commit fraudulent acts. The fact that Botnets keep continuously evolving means that traditional detection approaches are always one step behind. Recently, the behavior analysis of network traffic has arisen as a way to tackle the Botnet detection problem. The behavioral analysis approach aims to look at the common patterns that Botnets follow across their life cycle, trying to generalize in order to become capable of detecting unseen Botnet traffic. This work provides an analysis of the viability of Recurrent Neural Networks (RNN) to detect the behavior of network traffic by modeling it as a sequence of states that change over time. The recent success applying RNN to sequential data problems makes them a viable candidate on the task of sequence behavior analysis. The performance of a RNN is evaluated considering two main issues, the imbalance of network traffic and the optimal length of sequences. Both issues have a great impact in potentially real-life implementation. Evaluation is performed using a stratified k-fold cross validation and an independent test is conducted on not previously seen traffic belonging to a different Botnet. Preliminary results reveal that the RNN is capable of classifying the traffic with a high attack detection rate and an very small false alarm rate, which makes it a potential candidate for implementation and deployment on real-world scenarios. However, experiments exposed the fact that RNN detection models have problems for dealing with traffic behaviors not easily differentiable as well as some particular cases of imbalanced network traffic.**

***Resumen*—Una Botnet es un grupo de computadoras que pueden ser controladas de forma remota para ejecutar ataques coordinados o cometer fraudes. El hecho de que las Botnets esten evolucionando constantemente, hace que los enfoques tradicionales de detección esten siempre un paso por detrás. Recientemente, el análisis del comportamiento del tráfico de la red ha surgido como una manera de abordar el problema de detección de Botnets. El enfoque de análisis de comportamiento tiene como objetivo analizar los patrones comunes que una Botnet sigue en todo su ciclo de vida, tratando de generalizar a fin de llegar a detectar tráfico de Botnet no visto. En este trabajo se presenta un análisis de la viabilidad de aplicar Redes Neuronales Recurrentes (RNN) para detectar el comportamiento del tráfico de red. Para esto, el tráfico de la red es modelado como una secuencia de estados que cambian con el tiempo. El reciente éxito de la aplicación de RNN a los problemas de datos secuenciales hacen de estas un candidato viable a la tarea de análisis de comportamiento basado en secuencias. El rendimiento de un RNN es evaluado considerando dos problemas principales, el desequilibrio de tráfico de red y la longitud óptima de las secuencias. Ambos problemas tienen un gran impacto en una posible aplicación en la vida real. Los experimentos se realizan mediante una validación cruzada estratificada a lo que se agrega una prueba independiente sobre tráfico no visto anteriormente, el cual pertecene a una Botnet diferente. Los resultados preliminares revelan que las RNN son capaces de clasificar el comportamiento del tráfico con una alta taza de detección de ataques y una tasa de falsas alarmas casi insignificante, sin embargo, los experimentos expuestos en este trabajo exponen el hecho de que los modelos de detección basados en RNN presentan problemas para hacer frente a los comportamientos de tráfico que no son fácilmente diferenciables, así como algunos casos particulares de tráfico no balanceado.**

***Index Terms*—Traffic Behavior, Botnet Detection, Recurrent Neural Networks**

I. Introduction and Motivation

According to Hacheem et al. [1], a Botnet could be defined as a number of Internet computers that have been set up to establish connections to other computers on the Internet. The continuous evolving behavior of Botnets has caused that most of the traditional detection approaches failed to provide the necessary detection results [2]. Recently, the rise of behavioral detection approaches [3] have proved to be more adequate to deal with the constant change in the Botnet activities. A behavioral detection approach is based on finding the common patterns that Botnets follow across their life cycle, trying to generalize them in order to become capable of detecting unseen Botnet traffic. For instance, no matter what actions a Botnet has been ordered to perform, the fact is that periodically all the bots need connect to the Botmaster to receive new orders. Such kind of behaviors observed only after a long period of time is precisely the object of behavioral detection methods.

The Strastosphere [4] Intrusion Prevention System (**IPS**) project, is an initiative for providing to the community an state of the art behavioral IPS. The current Stratosphere IPS detection approach is based on first order Markov Models. Even thought the results have been promising, and improvements are continuously being developed, first order Markov models suffer from issues for memorizing a large state sequences. On the other hand, Recurrent Neural Network (RNN) have been proved to successfully deal with

large sequences, which makes them a viable candidate on the task of sequence behavior analysis. The present article discuss the application of RNN to detect the behavior of network traffic. In particular, an analysis of the application of a Large Short Term Memory (LSTM) network [5] for recognizing the different sequences of states that change over time. It is worth noticing that since the goal behind the Stratosphere project is to develop a fully functional IPS, the present work not only considers the detection performance of the LSTM detection models, but also the issues regarding the deployment of such type of Artificial Neural Network (ANN) on real network scenarios. In particular, two known problems are discussed: First, the capabilities of LSTM for dealing with imbalance network traffic. Second, the optimal length of state connections required for efficiently detecting traffic behaviors.

The rest of the article is organized as follows: Section II describes the strategy for generating the behavioral models from network traffic as it was proposed by the Stratosphere project. Then, the section III gives details about how a LSTM network has been adapted to perform behavior detection. In section IV, a description of the experiment design methodology followed for evaluating the performance of the LSTM detection model is provided. The results of evaluating the different sampling techniques and the optimal connection length are shown in sections VII and VII, respectively. An analysis of the results of a LSTM detection model on non previously seen data is presented in sections VII and VIII. Finally, the concluding remarks are exposed in section IX.

II. Behavioral Models

Behavioral models of malicious connections in the network are created by studying the long term characteristics of the network traffic. This kind of models have been implemented inside the Stratosphere Intrusion Prevention System (IPS) [6], which is a large effort for offering a state of art IPS for Non Governmental Organizations (NGO). The project involves a collaboration between two universities (CVUT and UNCuyo) together with other organizations.

The approach used by Stratosphere IPS to model the behavior of a connection, starts by aggregating the flows according to a 4-tuple composed of: the source IP address, the destination IP address, the destination port and the protocol. All the flows that match a tuple are aggregated together and referred as a *Stratosphere connection*. From a traffic capture several of these Stratosphere connections are created. Each one of the these Stratosphere connections contains a group of flows. The behavior of a connection is computed as follows:

1) Extract three features of each flow: size, duration and periodicity.
2) Assign to each flow a *state* symbol according to the features extracted and the assignment strategy shown in Table I.
3) After the assignment, each *connection* has its own string of symbols that represents its behavior in the network.

An example of these **behavioral models** is shown in Fig. 1. The figure shows the symbols representing all the flows for a Stratosphere connection based on UDP protocol from IP address 10.0.2.103 to port 53 of IP address 8.8.8.8.

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

Fig. 1. An example of the behavioral model of connection from IP address 10.0.2.103 to destination port 53 at IP address 8.8.8.8 port 53 using protocol UDP.

Without even considering any detection method, the behavioral models based on symbols shown in Fig. 1 have proved to be a good visualization approach for helping security analysts in their daily tasks.

For performing actual detections of malicious behavior, the current strategy followed by the Stratosphere IPS is to use a Markov Chain-based analysis of the transition probabilities from one symbol to the next [3]. Even though this approach has proved to be effective in several real-life scenarios, Markov Chain analysis suffers from an important limitation, i.e. the calculation of the transition probabilities for each state only depends on the previous state.

III. RNN Detection Models

LSTM networks are a special type of RNN first introduced by Hochreiter & Schmidhuber in 1997 [5]. LSTM networks can potentially become an improvement over the previous detection method based on Markov Models, since it is not necessary to predefine the number of states to analyze in the behavioral model sequence. For space reasons, a detailed explanation about LSTM networks has not been included in this article. The reader is referred to [5] for a complete explanation about LSTM.

The idea proposed in this work consists of using LSTM networks for building detection models based on the behavior of connections. The strategy used follows the classical supervised Machine Learning (ML) approach to build classifications models. Such approach is based on the use of historical data that have been previously labeled as *Normal* or *Botnet* and then train the LSTM network to finally obtain a detection model capable of recognizing connections behaviors.

The strategy used for encoding behavioral connections to LSTM input units consists of transforming each symbol into a binary vector. According to Table I there are 50

TABLE I
SYMBOL ASSIGNMENT STRATEGY FOR BUILDING BEHAVIORAL MODEL ACCORDING TO THE STRATOSPHERE PROJECT.

	Size Small			**Size Medium**			**Size Large**		
	Dur. Short	**Dur. Med**	**Dur. Long**	**Dur. Short**	**Dur. Med**	**Dur. Long**	**Dur. Short**	**Dur. Med**	**Dur. Long**
Strong Per	a	b	c	d	e	f	g	h	i
Weak Per.	A	B	C	D	E	F	G	H	I
Weak Non-Per.	r	s	t	u	v	w	x	y	z
Strong Non-Per	R	S	T	U	V	W	X	Y	Z
No Data	1	2	3	4	5	6	7	8	9

Symbols for time difference	
Between 0 and 5 seconds:	.
Between 5 and 60 seconds:	,
Between 60 and 5 mins:	+
Between 5 mins and 1 hour	*
Timeout of 1 hour:	0

possible symbols used in a behavioral model. Therefore, it is possible to represent each one of these symbols on a binary vector of size 50. Each element on the vector represents a symbol. Given a symbol, only the element representing such symbol will be active on the vector, whereas the rest remains inactive. An example of representation scheme can be observed in Eq. 1 where the vector for symbol *a* and in Eq. 2 that do the same for the symbol *b*

$$\mathrm{a} = [\ 1 \quad 0 \quad 0 \quad 0 \quad \cdots \quad 0 \quad 0 \quad 0\ ] \tag{1}$$

$$\mathrm{b} = [\ 0 \quad 1 \quad 0 \quad 0 \quad \cdots \quad 0 \quad 0 \quad 0\ ] \tag{2}$$

The complete sequence of the model behavior is encoded in the form of a matrix (see Eq. 3) where each row represents a different symbol. In addition, the matrix subscript indicating the row number is used to guarantee the correct time line of each symbol inside the behavioral connection. The matrix is then fed to the input units layer of the LSTM network one row at the time.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Two main issues are observed regarding the application of LSTM for building detection models.

The first issue is focused on the size and proportion of the historical data used to build the detection models. A known limitation of supervised ML approaches such as ANN is their need of balanced labeled data. In other words, it is required that the number of connection behaviors belonging to the Botnet class equals the Normal class. Under these imbalanced situations, it is possible that the learned model presents a bias to the majority class.

The second is related to the encoding of the symbols used for representing the behavioral models from Table I in a efficient way for LSTM input units. Even thought the LSTM model has the potential of dealing with an undefined sequence length, the fact is that at some point, the previous states of a behavioral model do not have a significant influence on the current state. Therefore, it is important to find the optimal length of connection states necessary to properly detect a malicious behavior and avoid the use of unnecessary computational time.

IV. Experiment Design

The proposed LSTM detection method is evaluated against two different datasets coming from network traffic captures taken from CVUT university campus networks. Both datasets are publicly available as part of the Malware Capture Facility Project (**MCFP**) [7].

Table II provides brief information about each of the two datasets. The first two columns show the Label used for referencing the dataset and a brief description of the Network behavior included in such group. Then, in column three and four, the number of connections labeled as Botnet as well as Normal. Finally, the last column shows the ID of each of the datasets in the MCFP. It is important to mention that in these datasets only connections based on the TCP protocol are considered. Notice that even though dataset A and B are from the same capture according the MCFP, they contain traffic of two different Botnets and also they shown a different class unbalance. While in dataset A Botnet Traffic surpasses Normal traffic, the opposite situation is observed in the case of dataset B.

TABLE II
General information about datasets

ID	Desc.	Botnet Conn.	Normal Conn.	MCFP IDs
A	**Bonet Neris**	2101	713	CTU13-42
B	**Bonet DonBot**	188	300	CTU13-47

Standard performance metrics for network detection evaluation are used for comparing the different approaches discussed. These metrics correspond to Attack Detection Rate (**ADR**) and False Alarm Rate (**FAR**). ADR is computed as the ratio between the number of correctly detected attacks and the total number of attacks. Whereas FAR rate is computed as the ratio between the number of normal connections that are incorrectly classified as attacks and the total number of normal connections.

The analysis of the LSTM model performance considers the two issues observed in section III. The class balance required for building a proper model detection as well as the optimal length of connection states required to feed the LSTM input layer.

To guarantee the independence of the results. Dataset A is used for training the LSTM network and selecting the optimal strategy for dealing with the LSTM issues previously mentioned. Whereas dataset B is used for testing the performance of the LSTM detection model on unseen connections

The implementation used in the experiments is based on the deep learning KERAS framework [8]. The LSTM hidden layer consists of 128 neurons and the network is trained using the *Resilient Backpropogation* algorithm with a *dropout* of 0.1 during 30 *epochs*. Such parameters have been obtained using a classical grid search strategy.

V. Sampling Techniques for dealing with imbalanced classes

The idea behind this experiment is to analyze the influence of well-known sampling techniques for dealing with imbalanced classes on LSTM performance. Specifically, two techniques are considered: **Undersampling** and **Oversampling**. Undersampling consists of randomly removing instances from the majority class in order to balance both classes. Whereas in the Oversampling technique, randomly selected instances from the minority class are duplicated until classes get even.

Both sampling techniques were applied to dataset A. An LSTM network was trained using stratified k-fold cross validation with k equals to 10. Given the randomness associated to both sampling techniques, the previous process was executed 50 times. Table III shows the average ADR and FAR values after 50 executions for both strategies. Results are compared with ADR and FAR values when no sampling strategy were used.

TABLE III
AVERAGE AND STANDARD DEVIATION VALUES FOR ADR AND FAR AFTER 50 EXECUTIONS FOR DIFFERENT SAMPLING STRATEGIES

	ADR		FAR	
	Avg.	*Sd.*	*Avg.*	*Sd.*
No Sampling	0.9796	0.0106	0.0372	0.0227
Under Sampling	0.9680	0.0197	0.0195	0.0179
Over Sampling	0.9601	0.0132	0.0111	0.0068

In general, the observed ADR values do not show a significant variation on the three evaluates cases. However, a minimal performance loss is observed when using the two sampling techniques. In the case of the Undersample technique, this behavior can be explained by the fact that less malicious connections have been used to train the LSTM network. Therefore, the obtained detection model may not have the opportunity to learn potentially valuable information from these lost connections. More difficult is to explain the results from the OverSampling technique, given that LSTM network has not suffer from any sample reduction. However, it is likely that the greater number of normal samples have influenced in the detection capability of the learned model. In other words, the bias to classify a new connection as malicious observed in the case of no sampling has been reduced since much more normal connections are included.

Regarding the FAR, an important reduction is observed when comparing both sampling techniques with the case when no sampling is applied. It seems that both sampling techniques have succeeded in reducing the tendency of classifying most of the connections as malicious observed in the no sampling case. The results showed no significant difference between both techniques, even though in average the Oversampling technique seems to perform better than undersampling.

In any case, given that the difference between Undersampling and Oversampling is not significant, the Undersampling technique is preferred since there is a considerable reduction of required data volume.

VI. ANALYSIS OF THE OPTIMAL LENGTH OF THE BEHAVIORAL MODELS

This second experiment focus on the exploration of the number of states in connections of the behavioral models in order to establish the optimal length required to feed the LSTM input layer. Table IV shows the results for an LSTM network varying the number of connection states used for building the detection model. As in experiment from section VII, a stratified k-fold cross validation with k equals to 10 were used for the evaluation. In this case, however, only the results of the Undersampling technique are reported.

TABLE IV
AVERAGE AND STANDARD DEVIATION VALUES FOR ADR AND FAR AFTER 100 EXECUTION CONSIDERING DIFFERENT NUMBER OF STATE CONNECTIONS

Number of States		**4**	**5**	**6**	**10**	**25**	**50**	**100**
ADR	***Avg***	0.953	0.955	0.962	**0.968**	**0.970**	0.968	0.969
	sd	0.024	0.025	0.022	0.022	0.019	0.021	0.021
FAR	***Avg***	0.023	0.021	0.020	**0.019**	**0.018**	0.016	0.020
	sd	0.017	0.018	0.018	0.017	0.016	0.014	0.039

As can been observed, the better average values for ADR are those for the case of 25 connection states. Beyond this point average values decrease. In the case of FAR, such improvement is observed with the case of 50 connection states. However, according to standard deviation values, there is no significant difference in terms of ADR and FAR when more than 10 connection states are considered. Except for the case of 100 connection states when performance seems to decrease.

To explain previous results, it is necessary to analyze the state frequency distribution of dataset A. Such state frequency distribution is shown in Fig. 2. There, it is possible to observe that the distribution shows a positive skew (i.e. the mass of the distribution is concentrated on the left of the figure). In general, connections with up to 25 states have an important representation in the dataset, however the majority is observed in those connections with state length up to 10. More interesting is the fact that most of the 6-length connections are labeled as Botnet, which turn them into a good pattern for discriminating between Normal and Botnet behaviors. The latter could be the explanation behind the lack of a significant variation in the results between connections of different length: the LSTM detection model need to consider just the first 6 states to be accurate and the remaining can be discarded by LSTM algorithm.

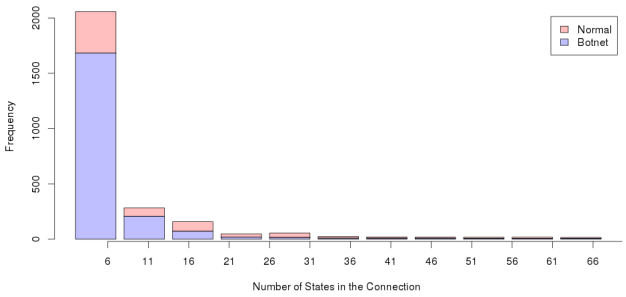

Fig. 2. Frequency histogram of the number of states per connection

To sum up, since the difference in performance between 10 and 25 states is negligible (it is about a 0.5% in ADR and 0.2% in FAR), a length of 10 states should be preferred. The latter choice responds to the reduction in the computational resources required for training the LSTM network.

VII. LSTM DETECTION MODEL APPLIED TO UNSEEN TRAFFIC

The idea behind this experiment was to analyze the influence of Undersampling and the selected state connection length on the performance of an LSTM detection model. LSTM detection model has been trained on dataset A using the selected sampling technique (Undersampling) and the optimal length of behavioral models (10). Then, the LSTM detection model was evaluated on dataset B.

Notice that to guarantee an independent and a fair evaluation, the LSTM detection model is trained in just one opportunity. This way, the carried out evaluation is as close as possible to a real-life situation.

The LSTM detection model tested on dataset B showed a value of **0.809** for ADR, while in the case of FAR the

observed value was **0.030**. As can be noticed, the LSTM detection model has suffered from a 15% of performance loss in terms of ADR and 1% in terms of FAR when compared with results shown in sections and .

An analysis of the detection performance of LSTM on the connections labeled as Botnet is shown in the Barplot of Fig. 3. The figure shows, for all the connections labeled as Botnet, the proportion of connections correctly classified (shown in color cyan) and the incorrectly classified (shown in color pink).

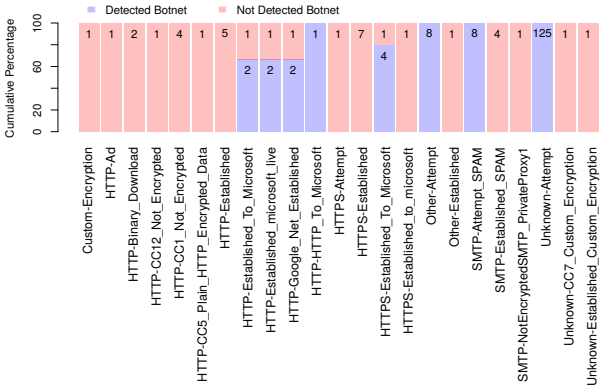

Fig. 3. Discriminative analysis of connections labeled as Botnet

The information provided by Fig. 3 exposes the fact that most of the Botnet traffic of dataset B come from *Unknown-Attempt* connections. An *Unknown-Attempt* refers to an attempt to establish a TCP connection on ports not commonly associated with standard services. As can be seen, the LSTM detection model is able to detect all the cases where this type of behavior is present, including those labeled as *Other-Attempt* and *SMTP-Attempt-SPAM*. The LSTM detection model has, however, failed to correctly identify most of the HTTP and HTTPS traffic (represented in the first 7 bars plotted in Fig. 3) as well as some of the traffic labeled as *Established* including *SMTP-Established-SPAM*.

The Fig. 4 shows the detection performance of LSTM on the connections labeled as Normal. In this case, most of the normal traffic in dataset B corresponds to HTTP or HTTPS connections and some other services such as Matlab server of Jabber. As can be seen, the LSTM detection model has correctly detected the majority of the normal traffic, with the exception of some of the HTTP traffic.

VIII. Discussion

To analyze the behavior of the LSTM detection model, first it is necessary to know the characteristics of the dataset used during the training process. Fig. 5 shows the labels frequency distribution for connection labeled as Botnet and normal. It is possible to observe that, for malicious connections, the most representative traffic corresponds to *SMTP-SPAM-Attempts* and attempts to establish an HTTP connection. There are also a number of connections labeled as *UNKNOWN*, which refers to traffic to some unknown port carrying an unknown protocol. In the case of normal connections, most of the traffic consists of HTTP and HTTPS to different sites.

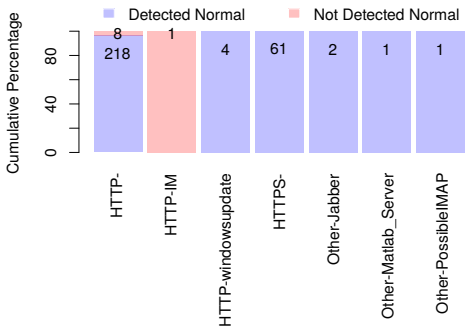

Fig. 4. Discriminative analysis of connections labeled as Normal.

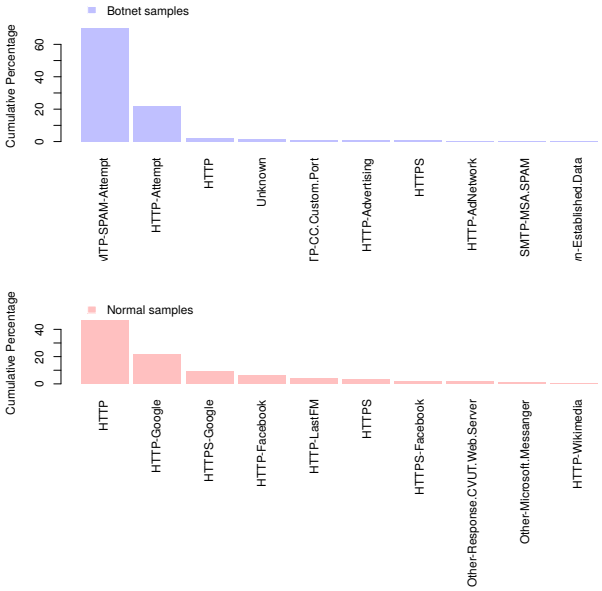

Fig. 5. Label frequency distribution of the 10 most representative connections for dataset A.

Connection attempts represent the large majority of the Botnet traffic used in dataset A. The fact that such type of traffic consists of only a few states and corresponds to the short connections shown in connection length histogram from Fig 2. These connections attempts are behind the lack of the significant variation in terms of ADR shown during the exploration of the length of behavioral models carried out in section VII. The influence of such kind of behaviors is strong and the remaining behaviors present in dataset A does not influence the final results. In addition, normal connections in dataset A do not include such kind of behaviors. Consequently, the LSTM has correctly detected a 99.9% of all the the connection attempts present in dataset B. Including SPAM, HTTP and UKNOWN attempts.

By contrast, the LSTM has failed in detecting most of the HTTP and HTTPS traffic. Such results could be explained by the imbalance situation observed regarding the labels of HTTP traffic. The majority of HTTP traffic is labeled as normal, while the number of HTTP connections labeled as Botnet are about the 11% of all the HTTP traffic present in dataset B. Under such situation the LSTM detection models tends to classify most of the HTTP traffic as normal.

In the case of FAR, it is important to mention that even with the presence of a majority of normal traffic on dataset B the value obtained is barely greater than those values obtained during analysis made in sections VII and VII. Actually, when considering standard deviations from table IV, the results do not show a significant difference.

A last observation regarding the results is related with the actual differences between connections belonging to both classes. There are behaviors that are clearly more distinguishable than others. This is the case of the connections attempts, which obviously represent some kind of abnormal behavior. No matter if such behaviour is caused by a Botnet or a normal connection, the fact is that a high number of connection attempts is a situation that any system administrator can easily recognize. A completely different situation is the case of HTTP established traffic. Certainly, such kind of connections are more difficult to recognize as malicious. An example of this can be seen in the histograms from Fig. 3, a connection labeled as *HTTP-Established* could be really difficult to distinguish from a normal HTTP connection.

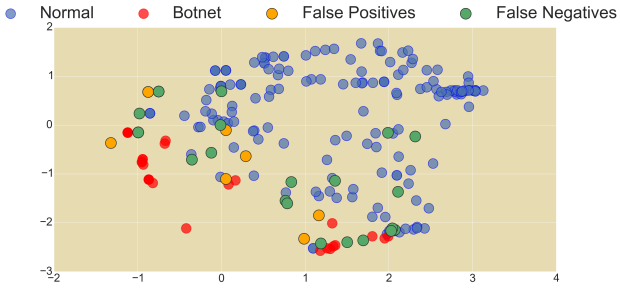

Fig. 6. 2D representation of Botnet and normal connections and the LSTM model detection classification for dataset B.

The plot from Fig. 6 is a 2D presentation of connections using Principal Component Analisys (PCA), which provides a way to analyze the similarity between different connections. Botnet connections are shown in red while normal are shown in blue. Additionally, those normal connections incorrectly detected as Botnet are shown in yellow while those Botnet connections not recognized are shown in green. In the figure it is possible to observe that there are many situations where Botnet and normal connections are very similar. A clear example of the latter situation can be observed in the quadrant (3,5) following a row-major order, where most of the connections are normal and two Botnet connections are very close to them. Under such level of similarity the LSTM detection model fails to correctly classify both Botnet connections. Another example is shown in the quadrant (3,1) where a Normal connection is incorrectly classified as Botnet. Such connection is close to a large set of Botnet traffic and far from normal connections. Therefore, at least under the current behavioral model, there will be traffic that could be difficult to differentiate.

IX. Concluding Remarks

The present work exposed the first steps in the implementation of the behavioral IPS based on LSTM. In particular, two main issues regarding the application of LSTM for detecting network behavior were analyzed. For the case of traffic imbalanced, Undersampling and OverSampling, two of the most common sampling strategies, were evaluated on a dataset with more Botnet than Normal traffic. Results have shown that if no sampling technique was used FAR value could increase considerably. Despite its simplicity, Oversampling resulted a more convenient sampling technique for improving detection. However, since no significant difference was observed between both sampling techniques, the Undersampling technique was preferred because it was more computationally efficient A similar computationally efficient approach was followed for establishing the optimal length of the state of connections, where a minimum of 10 states were selected for training the LSTM detection model.

A totally independent test was carried out on a unseen dataset with an opposite traffic imbalance situation (more normal than Botnet traffic) and a different type of Botnet. In general, the LSTM detection model has shown competitive values for ADR and FAR. Even the FAR values have remained inside the expected levels shown during cross validation. The latter is a very important result in a future real-life implementation.

A deeper analysis showed that LSTM was capable of detecting correctly those Botnet behaviors that were significantly different to Normal. On the other hand, when such differences were not obvious, the class containing most of the similar behavior was preferred (as is the case of HTTP protocol).

Finally, more experiments must be conducted in this direction, analyzing with more details other possible solutions regarding the per-connection imbalance situations. In particular, the case of normal traffic sampling. Additionally, a more effective way represent information to LSTM could help in differentiating traffic behaviors

Acknowledgments

The authors would like to thank the financial support received by UNCuyo and CVUT during this work. In particular the founding provided by Faculty Mobility program of the International Relations Department from UNCuyo. Finally, the financial support from SeCTyP-UNCuyo through project No. M004 is also gratefully acknowledged.

References

[1] N. Hachem, Y. Ben Mustapha, G. G. Granadillo, and H. Debar, "Botnets: Lifecycle and Taxonomy," in *Network and Information Systems Security (SAR-SSI), 2011 Conference on*. La Rochelle, France: IEEE, May 2011, pp. 1–8.

[2] C. A. Catania and C. G. Garino, "Automatic network intrusion detection: Current techniques and open issues," *Comput. Electr. Eng.*, vol. 38, no. 5, pp. 1062–1072, Sep. 2012.

[3] S. Garcia, "Identifying, Modeling and Detecting Botnet Behaviors in the Network," Ph.D. dissertation, UNICEN University, 2014.

[4] Straosphere IPS Project, "Stratoshpere Project," 2015. [Online]. Available: https://stratosphereips.org

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[6] S. Garcia, "Modelling the Network Behaviour of Malware To Block Malicious Patterns . The Stratosphere Project : a Behavioural Ips," in *Virus Bulletin*, no. September, 2015, pp. 1–8. [Online]. Available: https://www.virusbtn.com/pdf/conference_slides/2015/Garcia-VB2015.pdf

[7] Garcia, Sebastian, "Malware Capture Facility Project," 2013. [Online]. Available: https://mcfp.felk.cvut.cz/

[8] KERAS Development Team, "Keras: Deep Learning library for Theano and TensorFlow ," 2016. [Online]. Available: https://keras.io