An Analysis of Recurrent Neural Networks for Botnet Detection Behavior

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 the viability of Recurrent Neural Network (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 an stratified k-fold cross validation and a independent test is conducted on not previously seen traffic belonging to a different Botnet. Preliminary results reveal the RNN is capable of classifying the traffic with a high attack detection rate and an almost negligible false alarm rate, which makes it a a potential candidate for implementation and deployment on real-world scenarios. However, experiments exposed the fact that RNN detection models has problems for dealing with traffic behaviors not easily differentiable as well as some particular cases of imbalanced network traffic.**

Resumen—**Una Botnet consiste en un grupo de computadoras que pueden ser controladas de forma remota para ejecutar ataques coordinados o cometer actos fraudulentos. 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 ofrece 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 forward transmissions (usually malicious) to other computers on the Internet The continuous evolving behavior of Botnets has caused that most of the traditional detection approaches have shown unsuccessful results. Recently, the arise of behavioral detection approaches [2] 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 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 should connect to botmaster to receive new orders. Such kind of behaviors observed only after a long period of time is the precisely the object of behavioral detection methods.

The Strastosphere [3] Intrusion Prevention System (**IPS**) project, is an initiative for providing to the community an state of the art behavioral IPS. 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 Markov models suffer from issues for memorizing a large state sequences. On the other hand, Recurrent Neural Network (RNN) have proved to successfully deal with large sequences, which makes them a viable candidate on the task of sequence behavior analysis.

The present article focus on the capabilities of Recurrent Neural Network (RNN) to detect the behavior of network traffic. In particular, an analysis of the application of a Large Short Term Memory (LSTM) network [4] to recognize the different sequence of states that change over time. It is worth noting 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 network 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 detect traffic behaviors.

The rest of the article is organized as follows: Section II describes the strategy for generating the behavioral models

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 0 and 5 seconds:	*
Timeout of 1 hour:	0

from network traffic as was proposed by the Stratosphere project. Then, the section III gives details about how a Recurrent Neural Network has been adapted to perform behavior detection. In section IV, a description of the the experiment design methodology followed for evaluating the performance of the RNN 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 RNN detection model on non previously seen data is presented in sections VII and VIII. Finally, 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) [5], which is a large effort that 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 in a *connection*. From a traffic capture we can create several connections, each with its group of flows. For each flow in a connection it is applied the following steps to compute its behavior:

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) All the states (symbols) of the *connection* are represented as a string and stored as part of the behavioral model.

In the second step a symbol is assigned to each flow using the strategy shown in Table I. 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.

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 8.8.8.8-53 using protocol UDP

Without even considering any detection method, the behavioral models based on symbols shown in Fig. 1 has proved to be an good visualization approach for helping security analysts in their daily tasks. For performing actual detection of malicious behavior, the current strategy followed by the Stratosphere IPS is to use Markov Chain-based analysis of the transition probabilities from one symbol to the next [2]. Even tough this approach has proved to be effective in several real-life scenarios, the fact is that in Markov Chain analysis, each state can depend only on the previous state. In the following section we proceed to describe an special kind of Recurrent Neural Network (**RNN**) called **LSTM** (Long Short Term Memory). LSTM network have proved to be capable of building models that consider larger states dependencies in an computational efficient way [6].

III. RNN DETECTION MODELS

LSTM networks are a special type of RNN first introduced by Hochreiter & Schmidhuber in 1997 [4]. LSTM networks have efficient training algorithms which have facilitated their successful application on several problems. The main advantage of these kinds of RNN is their capabilities to deal with long term dependencies on large sequences in an autonomous way. In the context of behavioral models, such capability translates into the preservation of information about an undefined number of previous connection states. 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 [4] for 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 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 to finally obtain a detection model capable of recognizing connections behaviors.

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

$$a = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \quad (1)$$

$$b = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \quad (2)$$

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.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \quad (3)$$

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 Machine Learning 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 has 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 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 Bonet 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 TCP protocol are considered. Notice that even though dataset A and B are from the same capture according the MCFP, they contains traffic from a different Botnet and shown a different show 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 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. While dataset B is used for testing the performance of the LSTM detection model on unseen connections

The implementation used for the followed experiments is based on the deep learning KERAS framework [8]. The LSTM is trained using *Rprop* algorithm with a dropout of 0.1. 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 are applied to dataset A. An LSTM network is trained using stratified k-fold cross validation with 10 folds. Given the randomness associated to both techniques, the previous process is executed 50 times. Table III shows the average ADR and FAR values after 50 executions for when using both strategies. Results are compared with ADR and FAR values when no sampling strategy.

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 success in reducing the bias of classifying connections as malicious observed in the no sampling case. Results shows no significant difference between both techniques, even though in average the Oversampling sampling technique seems to perform better than undersampling.

In any case, given that the difference between Undersampling and Oversampling is not significant, the Undersampling technique should be 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 , a stratified k-fold cross validation with 10 fold is used for the evaluation. In this case, however, only 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

State number		4	5	6	10	25	50	100
ADR	*Avg*	0.9538	0.9556	0.9626	**0.9680**	**0.9702**	0.9680	0.9691
	sd	0.0241	0.0256	0.0228	0.0223	0.0195	0.0217	0.0212
FAR	*Avg*	0.0235	0.0211	0.0208	**0.0196**	**0.0180**	0.0168	0.0202
	sd	0.0177	0.0181	0.0182	0.0174	0.0163	0.0148	0.0390

As can been observed, the better average values for ADR are those for the case of 25 connection states. Beyond this point average values decay. In the case of FAR, such improvement is observed with the case of 50 connection states. However, if standard deviation values are considered, 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.

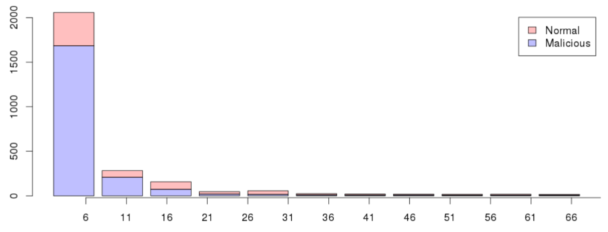

Fig. 2. Frequency histogram of the number of states per connection

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 is 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 is evaluated on dataset B.

Notice that to guarantee an independent and 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 shows a value of **0.809** for ADR, while in the case of FAR the observed value is **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 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 blue) and the incorrectly classified (shown in color green).

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 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

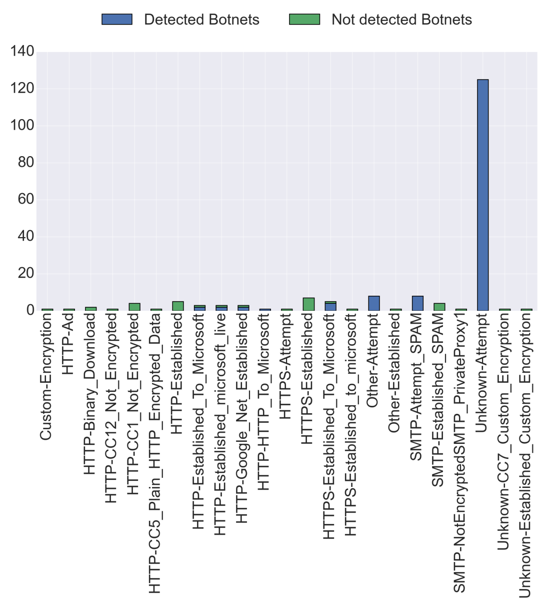

Fig. 3. Discriminative analysis of connections labeled as Botnet

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

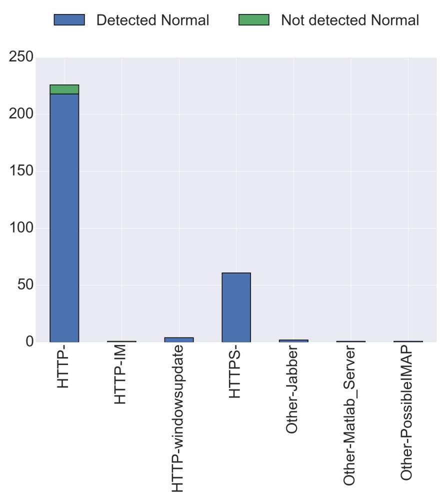

Fig. 4. Detections between groups

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.

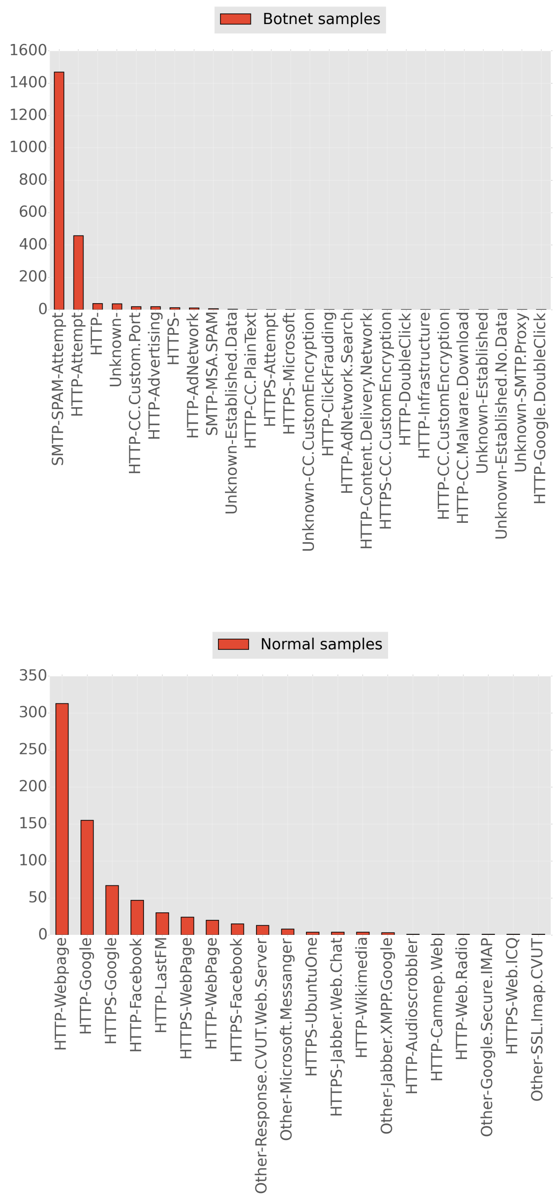

Fig. 5. Label frequency distribution for dataset A

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 does 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, result 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 that 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 shows a 2D representation of the connections present in dataset B. Botnet connections are shown in red while normal are shown in blue. Additionally, those normal connection incorrectly detected as Botnet are shown in yellow while those Botnet connections not recognized are shown in green.

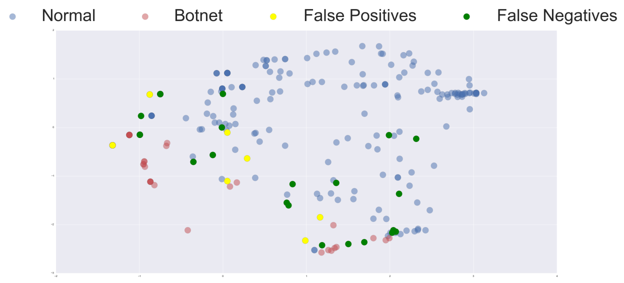

Fig. 6. 2D representation of Botnet and normal connections and the LSTM model detection classification for dataset B

The plot from Fig. 6 provides a way to analyze the similarity between different connections. There, 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 connection 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 (described in section II), it is very difficult to differentiate.

IX. Concluding Remarks

Two main issues regarding the application of LSTM for detecting network behavior are analyzed. For the case of traffic imbalanced, two of the most common sampling strategies are evaluated on a dataset with more Botnet than normal traffic. Results have shown that if no sampling technique is used FAR value could be a major issue. Despite its simplicity Oversampling seems more convenient for improving results. However, in this work Undersampling technique is preferred. The latter, responds to the fact that the goal of this work is to bring closer the gap between IPS research and it application on real life scenarios. A similar approach was followed for establishing the optimal length of the state connections, where a maximum of 10 states are used for training the LSTM detection model.

A totally independent test is carried out on a dataset with a the opposite traffic imbalance (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 shows that LSTM is capable of detecting correctly those Botnet behaviors that are significantly different to normal. On the other hand, when such difference is not obvious, the class containing most of the similar behavior is preferred (as is the case of HTTP protocol).

More research 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

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

[3] ——, “Stratoshpere Project,” 2015. [Online]. Available: https://stratosphereips.org

[4] 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

[5] 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

[6] Z. Lipton, J. Berkowitz, and C. Elkan, “A Critical Review of Recurrent Neural Networks for Sequence Learning,” *arXiv preprint*, pp. 1–35, 2015. [Online]. Available: http://arxiv.org/abs/1506.00019v2

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

[8] K. D. Team, “Keras: Deep Learning library for Theano and TensorFlow ,” 2016. [Online]. Available: https://keras.io