$See \ discussions, stats, and author \ profiles \ for \ this \ publication \ at: \ https://www.researchgate.net/publication/340539486$ 

# Neural Network Architectures for the detection of SYN flood attacks in IoT systems

## Preprint · April 2020

DOI: 10.13140/RG.2.2.21280.33287

CITATION: 0	5	READS 127					
0		121					
4 autho	rs, including:						
	Erol Gelenbe						
	Polish Academy of Sciences						
	784 PUBLICATIONS 18,730 CITATIONS						
	SEE PROFILE						
Some of the authors of this publication are also working on these related projects:							
Project	Emergency Management Systems View project						

Project GHOST - Safe-Guarding Home IoT Environments with Personalised Real-time Risk Control View project

## Neural Network Architectures for the detection of SYN flood attacks in IoT systems

Spilios Evmorfos<sup>†</sup> Internet Science Group Institute of Communication and Computer Systems Athens, Greece George Vlachodimitropoulos Internet Science Group Institute of Communication and Computer Systems Athens, Greece Nikolaos Bakalos National Technical University of Athens Athens, Greece

Erol Gelenbe IITIS Polish Academy of Science Gliwice, Poland

## ABSTRACT

We investigate light-weight techniques for detecting common SYN atttacks on devices that are attachd to the Internet, such as IoT devices and gateways, Fog servers or edge devices which may have low processing capacity. In particular we examine the Random Neural Network with Deep Learning, trained with "normal" non-attack traffic, and a Long-Short-Term-Memory (LSTM) neural network. Using the same traffic traces for attack traffic, our experiments show that the Random Neurla Network provides substantially better attack detection and significantly lower false alarm rates as compared to the LSTM network.

## **KEYWORDS**

Neural networks, cybersecurity, classifiers of IoT flows, Lightweight attack detection

## 1 Introduction

Network security has always been in the forefront of networking-related research. The focus has previously been on the security aspects of traditional TCP/IP networks, but the rise of IoT (Internet of Things) networks results in the emerging of a new landscape in terms of security.

The category of attacks, most typical in traditional TCP/IP networks, is the one related to the interception of valuable information. On the other hand, in IoT networks, the attacks that are most common and least explored, are those labeled as Denial of Service (DoS) attacks. In that particular type of attacks, the attacker attempts to inhibit the target's ability to function seamlessly.

In this paper, we exploit the immense modelling capabilities of two different types of deep neural networks: The Long-Short-Term-Memory (LSTM) and the Random Neural Network, for Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s). *WOODSTOCK'18, June, 2018, El Paso, Texas USA* 

© 2018 Copyright held by the owner/author(s). 978-1-4503-0000-0/18/06...\$15.00 https://doi.org/10.1145/1234567890

detecting a common type of DoS attack, the SYN flood attack. The two neural network architectures represent two different formulations. The LSTM is a recurrent formulation and the Random Neural Network is implemented as feed forward (even though the Random Neural Network architectures can also be recurrent).

#### **1.1 Previous Work**

The immense capabilities of neural networks to extract complex patterns from given data intuitively seems a great tool to use for detecting malicious activities in the context of an IoT network. LSTMs are renowned for applications of handling multivariate time series [8] and in general cases where the data intrinsically show some temporal dependencies. On the other hand, Random Neural Networks [1][2][3] seem to have a broader spectrum of possible applications.

The SYN flood attack has been described in [4] and is a basic type of Denial of Service attack. The attacker exploits the SYN TCP 3-WAY handshake and initiates many connections with the same port of the target but establishes none. So, the attacker renders the target's ability to handle new requests.

Deep learning has been used before for the detection of SYN flood attacks in [4] where a Random Neural Network was implemented as a classifier to distinguish between non-malicious network packet captures and captures constituting SYN attacks.

The LSTM neural network architecture has also been previously implemented for detecting DoS attacks on networked infrastructures[6], using a Bayesian inference approach rather than via direct learning based on the traffic patterns. The intersection between deep learning and detection of Distributed Denial of Service (DDoS) attacks was also investigated in [5] and showed the effectiveness of deep neural networks in modelling the attackers' patterns in attempting to perform DoS attacks. Deep learning has also been employed in security for its ability to extract the patterns in attack sequences, and is reviewed in [7].

Deep learning and deep neural networks have been implemented not only for attack detection, but in the whole spectrum of assisting the task of securing IoT systems. In [8] a secure routing method using Random Neural Network architectures for decision making in SDN Controllers has been presented, and [18] discusses similar schemes for task allocation in Cloud servers.

## **1.2 Our Contribution**

In this paper we simulate SYN TCP flood attacks in IoT systems. We use this simulation to train machine learning modules, as time series predictive models to be used in detecting these particular types of attacks and differentiate them from normal, "malign" traffic flows. We compare the performance of different neural network architectures in their ability to track the distribution of normal traffic.

## 2 Neural Network Structure

## 2.1 Random Neural Network

The Random Neuron is a unit that receives two types of input signals, the excitatory and the inhibitory and is also characterized by its rate that is always positive. If we denote as x the excitatory input, as y the inhibitory input and as r the rate, the output of the Random Neuron is  $z = \min \left\{ \frac{x}{r+y} \right| 1 \right\} r$ 

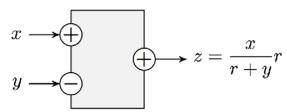


Figure 1: Structure of a Random Neuron

In the feedforward formulation of the Random Neural Network there are no circuits in the connection graph. There are three distinct categories of layers, the input layer, the hidden layers and the output layer. Every unit is connected to other units that belong to the hierarchically consecutive layer (from the input layer to the output layer passing through the hidden layers). This formulation results to a non-linear system of equations that can be formally solved [9].

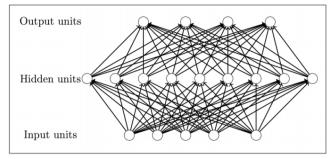


Figure 2: Single Layer RNN

Let I be the number of neurons in the input layer, H is the number of neurons in the hidden layer (assuming there is only one hidden layer in the topology) and O is the number of neurons in the output layer. We provide index for every neuron in the feedforward formulation with the following methodology. We index the neurons of the input layer from 1 to I, the hidden neurons from I+1 to I+H and the output neurons from I+H+1 to I+H+O = N. Assuming that the input neurons are the only ones receiving signals from the outside we can compute the rates of activity for all the neurons:

• 
$$\rho_{k} = \frac{\lambda_{k}^{+}}{r_{k} + \lambda_{k}^{-}} \quad 0 \leq \kappa \leq I$$
  
• 
$$\rho_{h} = \frac{\sum_{k=1}^{I} \rho_{k} w_{kh}^{+}}{r_{h} + \sum_{k=1}^{I} \rho_{k} w_{kh}^{-}} I + 1 \leq h \leq H + I$$
  
• 
$$\rho_{o} = \frac{\sum_{h=I+1}^{I+I} \rho_{h} w_{ho}^{+}}{r_{o} + \sum_{h=I+1}^{I+I} \rho_{h} w_{ho}^{-}} I + H + 1 \leq o \leq N$$

As it has been shown in [2], the original gradient descent iterative optimization scheme can be tweaked and implemented for training feed forward neural network architectures both as regressor and as classifiers.

#### 2.2 Long-Short-Term-Memory (LSTM)

Long Short-Term Memory (LSTM) networks, as a special structure of Recurrent Neural Networks, have proven to be stable and powerful for modeling long-range dependencies in generalpurpose sequence modeling ([25]-[27]). In LSTMs, each node in the hidden layer is replaced by a memory cell, instead of a single neuron [25]. The structure of a single memory cell is depicted in the figure below.

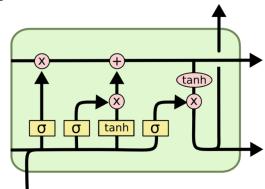


Figure 3: Structure of an LSTM Memory Cell

The memory cell contains the following components: the forget gate, the input node, the input gate, and the output gate. Each component applies a non-linear relation on the inner product between the input vectors and respective weights (altered iteratively through a training process). Some of the components have the sigmoid function,  $\sigma(\cdot)$  and others the tanh( $\cdot$ )

As discussed in [10] Recurrent neural networks and LSTMs in particular, have shown great success in predicting time series online. Especially in [11] LSTMs have been used to tested, particularly on predicting traffic flows.

The goal of the forget gate is to decide what information should be discarded out of the memory cell [12]. The output, denoted as f(n) ranges between 0 and 1, according to the sigmoid activation function. The forget gate learns whether a previous or future vector state is necessary for the estimation of the current value state. The input node performs the same operation with that of a hidden neuron of a typical recurrent regression model. The goal of this node is to estimate the way in which each latent state variable contributes to the final model.

As far as the input gate is concerned, its role is to regulate whether the respective hidden state is sufficiently important. It has the sigmoid function, therefore its response ranges between 0 and 1. This gate addresses problems related to the vanishing of the gradient slope of a  $tanh(\cdot)$  operator. Finally, the output gate regulates whether the response of the current memory cell is sufficiently significant to contribute to the next cell. Therefore, this gate actually models the long-range dependency together with the forget gate.

The recurrent nature of the LSTM presents many intricacies in terms of the iterative training process for adjusting the weights of the multiple gates. The adaptation of the backpropagation algorithm for accommodating the LSTM training is called Backpropagation Through Time [13]. The backpropagation variation for training recurrent neural network architectures presents the problem of vanishing or exploding gradients. So the number of time steps that the gradient is propagated is another hyperparameter of training that needs to be monitored. This adaptation is called truncated backpropagation through time and is thoroughly explained in [14].

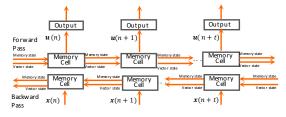


Figure 4: LSTM Architecture

## 3 System Architecture

The basic premise of the methodology for detection is described below. The Communication in the context of a network is captured in a pcap file using Wireshark [15]. The communication contains both non malicious traffic and SYN flood attacks targeted towards the port of a specific node.

The pcap file is used for creating an annotated dataset and being made into a univariate time series. Specifically, the pcap is being dissected into time windows of 5 seconds. During the period of 5 seconds, special Wireshark filters were used to count the number of half opened TCP connections established with a specific port of a particular IP during the time frame. In that way the final dataset is a univariate list of the number of unestablished TCP connections.

The basic idea is to use a deep neural network as a regressor and train it with a part of the time series that corresponds to normal non malicious communication. Then the a priori trained neural network regressor attempts to predict the number of half-open TCP connections for the consecutive time window. If this number deviates from the actual value of the metric by a predefined threshold then the inspected node is considered to be under attack.

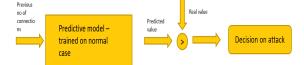


Figure 5: Workflow of the proposed method

#### LSTM architecture:

The LSTM neural network architecture is comprised by one input layer, one output layer and two hidden layers with 50 neurons each (dense formulation). The Loss function used for adapting the weights is the Mean Square Error (MSE) which is the most typical loss function used for training in regression problems [16] and the optimization scheme is the ADAM optimizer [17]. The Backpropagation Through Time (BPTT) was stopped at three consecutive steps going back so the truncated version of the Backpropagation scheme was implemented for avoiding vanishing gradients.

#### Random Neural Network architecture:

The Random Neural Network was in feedforward formulation so no recursive element. Other than the input and output layers, there was one hidden layer with 50 neurons. The nature of the Gelenbe Networks entails no choice for the activation function. The loss function was again the Mean Square Error function and for the iterative optimization scheme, the adaptation of the backpropagation scheme as described in [2] was implemented from scratch (without using any high-level API implementation)

## **4** Performance Evaluation

#### 4.1 Dataset Description

A bot network was created in lab environment. Every Virtual Machine (VM) simulated a node in the IoT network. Scapy (a python package for packet crafting and manipulation) was used to create a script that runs on every VM and creates TCP connections with the targeted node (simulates a possible server under attack).

Scapy was also used to create a script that manifests a SYN TCP attack towards the server. The script initiates multiple TCP connections from multiple ports of the attacker with a particular port of the destination. The connections are never fully established.

The whole communication is captured in pcap files using Wireshark which is a tool for network traffic monitoring. Even though, the communication in the context of the network, is nonmalicious, for the most part, the attack is being launched at specific instances of the duration of the experiment.

The pcap files are annotated with the methodology described in the previous section.

## 4.3. Experimental Validation

We have conducted experiments to: 1) validate the efficacy of the deep learning predictive model idea for SYN TCP attack detection and 2) compare the two architectures of deep neural networks in terms of accuracy.

We train each of the formulations of deep neural networks (always as a regressor) with the same dataset that has been derived from the annotation process of a pcap file that contains only nonmalicious communication.

Then we test the accuracy of the models by using the previously described methodology on a dataset that combines non malicious and malicious communication. We present the results

Table 1: Performance metrics for deep learning predictive models. The proposed Random Neural Network approach outperforms the LSTM

**Table 2: Comparative Performance of the model** 

Neural Network architecture	Accuracy	False Positives
Gelenbe-		
Network	80.7%	19.3%
LSTM	62.7%	37.3%

Here we should note that the formulation of the model architecture intuitively excludes the presence of False negatives and that is also prevalent in the results presented.

#### 5 Conclusion

In this paper we propose that a deep neural network formulated as a time series predictive model, that is trained under normal nonattack conditions, be used as a detection method for SYN TCP flood attacks in IoT systems (especially on lightweight IoT devices). Especially we propose the implementation of a Random Neural Network (Gelenbe Network) as the architecture for the deep neural network for the predictive model (especially on a feed forward formulation).

The Random Neural Network was observed to be better at capturing the boundaries between the different modes of the distribution of normal traffic, and its architecture appears to be more effective in detecting attacks, rather than at finding outliers, as compared to in comparison to a conventional sigmoidal deep neural network. In future work, the Random Neural Network's recurrent structure may also be used to improve its effectiveness even more as compared to a conventional deep learning approach based on feedforward neural architectures.

## ACKNOWLEDGMENTS

This research was partially supported by funding from the H2020-IOT-2016-2017 (H2020-IOT-2017) Program under Grant Agreement 780139 for the SerIoT Research and Innovation Action.

## REFERENCES

- [1] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution,"Neural Computation, 1(4), 502–510, 1989.
- [2] E. Gelenbe, "Learning in the recurrent random neural network.," Neural Computation, 5(1), 154–164, 1993
- [3] E. Gelenbe and Y. Yin, "Deep learning with random neural networks," Neural Networks (IJCNN), 2016 Int. Joint Conference on, IEEE, 1633–1638, 2016.
- [4] O. Brun, Y. Yin and E. Gelenbe, "Deep Learning with Dense Random Neural Networks for detecting attacks against IoTconnected home Environments," Procedia Computer Science, 134, 458-46 2018.

https://doi.org/10.1016/j.procs.2018.07.183

- [5] X. Yuan, C. Li and X. Li, "DeepDefense: Identifying DDoS Attack via Deep Learning," 2017 IEEE International Conference on Smart Computing (SMARTCOMP), Hong Kong, 2017, pp. 1-8. doi: 10.1109/SMARTCOMP.2017.7946998
- Y. Li and Y. Lu, "LSTM-BA: DDoS Detection Approach Combining LSTM and Bayes," 2019 Seventh International Conference on Advanced Cloud and Big Data (CBD), Suzhou, China, 2019, pp. 180-185. doi: 10.1109/CBD.2019.00041
- [7] T. Guo, Z. Xu, X. Yao, H. Chen, K. Aberer and K. Funaya, "Robust Online Time Series Prediction with Recurrent Neural Networks," 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, QC, 2016, pp. 816-825.
- [8] E. Gelenbe, "A Software Defined Self-Aware Network: The Cognitive Packet Network," 2014 IEEE 3rd Symposium on

Network Cloud Computing and Applications (ncca 2014), Rome, 2014, pp. 9-14.

- [9] S. Basterreich and Gerardo Rubino. "Random Neural Network Model for Supervised Learning Problems: A Tutorial," Neural Network World 25.5 (2015): 457–499. Crossref. Web.
- [10] T. Guo, Z. Xu, X. Yao, H. Chen, K. Aberer and K. Funaya, "Robust Online Time Series Prediction with Recurrent Neural Networks," 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, QC, 2016, pp. 816-825
- [11] Y. Tian and L. Pan, "Predicting Short-Term Traffic Flow by Long Short-Term Memory Recurrent Neural Network," 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), Chengdu, 2015, pp. 153-158.

doi: 10.1109/SmartCity.2015.63D.I.,.

- [12] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink and J. Schmidhuber, "LSTM: A Search Space Odyssey," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222-2232, Oct. 2017. doi: 10.1109/TNNLS.2016.2582924.
- P. J. Werbos, "Backpropagation through time: what it does and how to do it," in *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550-1560, Oct. 1990. doi: 10.1109/5.
- [14] H. Tang and J. Glass, "On Training Recurrent Networks with Truncated Backpropagation Through time in Speech Recognition," 2018 IEEE Spoken Language Technology Workshop (SLT), Athens, Greece, 2018, pp. 48-55. doi: 10.1109/SLT.2018.8639517
- [15] S. Kakuru, "Behavior based network traffic analysis tool," 2011 IEEE 3rd International Conference on Communication Software and Networks, Xi'an, 2011, pp. 649-652.

doi: 10.1109/ICCSN.2011.6014810

- [16] W. Zhang et al., "LSTM-Based Analysis of Industrial IoT Equipment," in *IEEE Access*, vol. 6, pp. 23551-23560, 2018. doi: 10.1109/ACCESS.2018.2825538
- [17] Z. Zhang, "Improved Adam Optimizer for Deep Neural Networks," 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS), Banff, AB, Canada, 2018, pp. 1-2. doi: 10.1109/IWQoS.2018.8624183
- [18] Lan Wang and E. Gelenbe, "Adaptive dispatching of tasks in the cloud," IEEE Transactions on Cloud Computing 6 (1), 33-45, 2018.