Deep Learning with Dense Random Neural Network for Detecting Attacks against IoT-connected Home Environments

Olivier Brun, Yonghua Yin, Erol Gelenbe

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2018) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

First workshop on Secure and Efficient Deployment of IoT
(SEDIT 2018)

Deep Learning with Dense Random Neural Network for Detecting Attacks against IoT-connected Home Environments

Olivier Brun[a,b,*], Yonghua Yin[b], Erol Gelenbe[b]

[a] *LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France*
[b] *Imperial College, London SW7 2AZ, UK*

Abstract

In this paper, we analyze the network attacks that can be launched against Internet of Things (IoT) gateways, identify the relevant metrics to detect them, and explain how they can be computed from packet captures. We then present the principles and design of a deep learning-based approach using dense random neural networks (RNN) for the online detection of network attacks. Empirical validation results on packet captures in which attacks are inserted show that the Dense RNN correctly detects attacks. However our experiments show that a simple threshold detector also provides results of comparable accuracy on the same data set.

Keywords: Cybersecurity; IoT; deep learning; dense random neural network.

1. Introduction

With the proliferation of network attacks aiming at accessing sensitive information without authorisation, or at rendering computer systems unreliable or unusable, cybersecurity has become one of the most vibrant of today research areas. Whereas most work has been done in the context of traditional TCP/IP networks, IoT systems have specific vulnerabilities which need to be addressed. In this paper, we analyze the cybersecurity threats against an IoT-connected home environment and present the principles and design of a learning-based approach for detecting network attacks.

As shown in Fig. 1, our attack detection approach relies on the analysis of the traffic flows exchanged with the IoT gateway. The data packets exchanged with the IoT gateway are captured on all network interfaces. These packet flows are then analyzed in order to extract various packet-level metrics from which network attacks can be detected. A classification algorithm, which has been previously trained with ”normal” IoT traffic, takes as input these metrics and predicts the probability that the IoT-connected home environment is currently under attack.

* Corresponding author. Tel.: +33-056-133-6918 ; fax: +33-056-133-6300.
E-mail address: brun@laas.fr

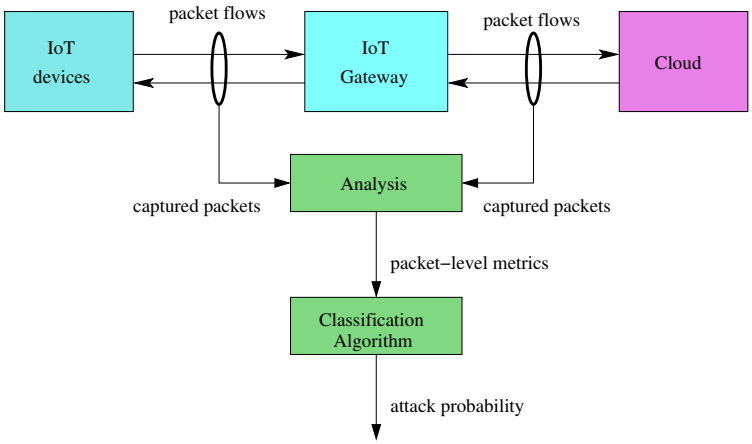


Fig. 1: Architecture of the online attack detection system..

The paper is organized as follows. In Sec. 2, we analyze the vulnerabilities of IoT gateways, identify the relevant metrics for detecting some of the attacks against them and explain how these metrics can be extracted from packet capture files. Sec. 3 is devoted to the description of the learning algorithm, whereas Sec. 4 presents empirical validation results. Some conclusions are drawn in Sec. 5.

2. Network Attacks

In an IoT-connected home environment, there may be dozens or even hundreds of sensors with various functions (e.g., measuring temperature, light, noise, etc) as well as some actuators for controlling systems such as the heating, ventilation, and air conditioning system. Each of these devices may use different communication protocols to connect (Wi-Fi, Bluetooth, Ethernet, ZigBee and others) and most of them are not able to connect directly to the Internet. A crucial component is then the IoT gateway, which is a device capable of aggregating and processing sensor data before sending it to Internet servers for further processing. A comprehensive project addressing the security of IoT gateways and related support systems in the Cloud is described in [19].

As IoT gateways sit at the intersection of edge devices (sensors and actuators) and the Internet, they are vulnerable to both traditional IP attacks and to attacks against wireless sensor networks. We consider both types of attacks. As there is a myriad of different computer and network attack methods, we focus on some of the most common and most damaging ones:

- **Denial-of-Service Attacks**: A denial-of-service (DoS) attack is typically accomplished by flooding the targeted machine or resource with superfluous requests in an attempt to overload systems and prevent some or all legitimate requests from being fulfilled [1, 2]. In a distributed denial-of-service attack (DDoS attack), the incoming traffic flooding the victim originates from many different sources, making it impossible to stop the attack simply by blocking a single source. Well-known examples of DoS attacks include *TCP SYN attacks*, *UDP and ICMP floods* and *HTTP POST DoS attacks*.
- **Denial-of-Sleep attacks**: In the context of the Internet of Things, low-rate wireless personal area networks are a prevalent solution for communication among devices. As discussed in [4], tights limitations on hardware cost, memory use and power consumption have given rise to a number of security vulnerabilities, including traffic eavesdropping, packet replay, and collision attacks, straightforward to conduct. A simple form of attack is to deplete the energy available to operate the wireless sensor nodes [6, 8]. For instance, *vampire attacks* are routing-layer resource exhaustion attacks aiming at draining the whole life (energy) from network nodes, hence their name [13]. Another type of energy attacks is known as Denial-of-Sleep attacks. These MAC-layer attacks can take on a variety of forms: *Sleep Deprivation attacks* [11, 12, 7], *Barrage attacks*, *Broadcast attacks* [3], *Synchronization attacks* [10], *Replay attacks* [5], and *Collision attacks* [9].

Table 1: Selected attacks and relevant metrics to detect them.

Attack	Metric
UDP flood	Number of outgoing ICMP "destination unreachable" packets
TCP SYN	Number of half-opened TCP connections
Sleep Deprivation Attack	Number of data packets over a long time scale
Barrage Attack	Number of data packets over a short time scale
Broadcast Attack	Number of broadcast messages

By carefully analyzing the principles of these attacks, it is possible to identify the metrics from which they can be computed. Tab. 1 presents the relevant metrics for detecting some of the attacks described above. These metrics can be computed from packet capture files using Scapy, which is a packet manipulation tool for computer networks, written in Python by Philippe Biondi. It can forge or decode packets, send them on the wire, capture them, and match requests and replies. It can also handle tasks like scanning, tracerouting, probing, unit tests, attacks, and network discovery. In our case, Scapy was used to produce the time-series associated to various network metrics from pcap files, and the resulting data sets were in turn used to train the classification algorithm.

3. Network-attack detection with the random neural network

This section describes the use of random neural networks (RNN) [17, 18] developed for deep learning recently [16, 14, 15] to detect network attacks, which can be viewed as a binary classification problem. First, we show how to construct training datasets from captured packets. Then, the dense RNN is presented to learn given datasets so as to conduct classification.

3.1. Dataset construction

Starting with the captured packets, statistical data (e.g., the number of UDP ports opened per time slot) in time series can be obtained as explained in Sec. 2. We extract samples from the time-series statistical data by setting a sliding window with length l. If a sample $X_n \in R^{l\times 1}$ is extracted in the non-attack case, then we assign the label of this sample denoted as y_n as 0; otherwise, if it is extracted in the attack case, the label of this sample is assigned as $y_n = 1$. Then, we have a dataset $\{(X_n, y_n)|n = 1, \cdots, N\}$, where the input is the statistical data extracted from captured packet data and the output is a binary value.

3.2. Dense random neural network with deep learning

A dense cluster in a "Dense RNN" was introduced in [16, 14] to mimic densely packed ganglia. The model is a standard RNN composed of n statistically identical cells in the case where the number of cells is very large. Each cell receives inhibitory spike trains from external cells with rate x, whose connection topology is a random graph with equi-probable connections to the set of all other cells.

Let q denote the probability of the activation state of a cell in the cluster in steady state. Previous work shows that a numerical solution can be obtained for q such that

$$q = \zeta(x) = \frac{-(c - nx) - \sqrt{(c - nx)^2 - 4p(n-1)(\lambda^- + x)d}}{2p(n-1)(\lambda^- + x)},$$

with $d = n\lambda^+$ and $c = \lambda^+ p + rp - \lambda^- n - r - \lambda^+ pn - npr$, where p is the repeated-firing probability when a cell fires, r is the firing rate of a cell, and a cell receives excitatory and inhibitory spikes from external world with rates λ^+ and λ^- respectively. For notation ease, $\zeta(\cdot)$ is used as a term-by-term function for vectors and matrices.

A Dense RNN in a multi-layer architecture (DenseRNN) is constructed in the following manner. The first layer (input layer) of the DenseRNN is made up of RNN cells that receives excitatory spike trains from external sources,

resulting in a quasi-linear cell activation $q(x) = \min(x, 1)$. The successive L layers are hidden layers composed of dense clusters that receive inhibitory spike trains from cells in the previous layer, with a resultant activation function $q(x) = \zeta(x)$. The last layer is an RNN-ELM. Let us denote the connecting weight matrices between layers of a L-hidden-layer ($L \geq 2$) DenseRNN by $W_1, \cdots, W_L \geq 0$ and output weight matrix by W_{L+1}. Given input matrix X, a forward pass of X in the DenseRNN can be described as:

$$\begin{cases} Q_1 = \min(X, 1), \\ Q_l = \zeta(Q_{l-1} W_{l-1}) \text{ for } l = 2, \cdots, L+1, \\ O = Q_{L+1} W_{L+1}. \end{cases}$$

where Q_1 is the 1st layer output, Q_l is the lth layer output ($l = 2, \cdots, L+1$) and O is the final DenseRNN output.

Given a training dataset $\{(X_n, y_n) | n = 1, \cdots, N\}$, the work in [16, 14] has developed an efficient training procedure for a DenseRNN to determine the values of $W_1, \cdots, W_L, W_{L+1}$, combining unsupervised and supervised learning.

4. Experimental Results

4.1. Experiment Setup

Some packet captures were obtained from a standard installation of the Carelife system. The Televes gateway was connected to the Internet using a 3G SIM card. Several software modules were installed on the gateway in order to capture and parse (in a PCAP file format) the data packets exchanged with various sensors which were previously paired and registered by the gateway, as well as those exchanged by the gateway with Internet servers. Packets were captured for a complete weekend on all the network interfaces of the gateway (see Fig. 2).

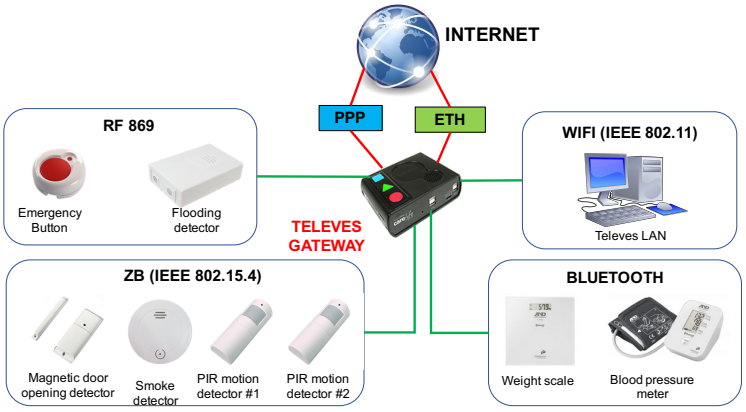


Fig. 2: Configuration used for the experiment.

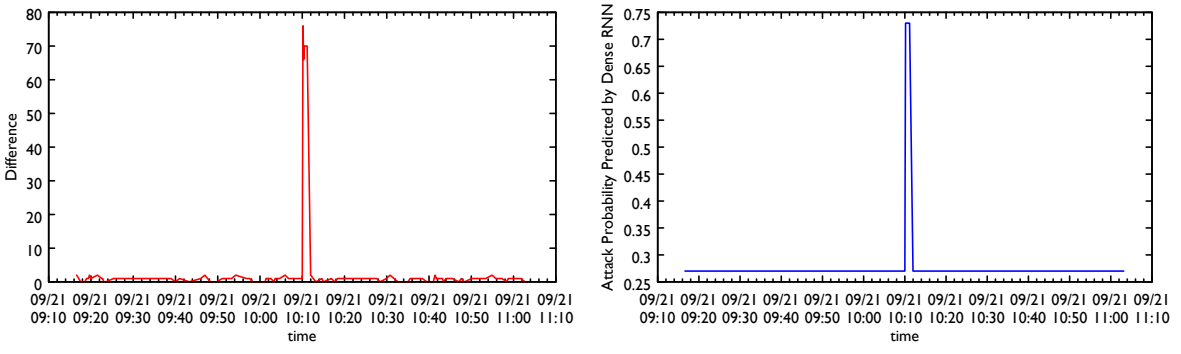
In the following, we shall focus on the packets captured on the PPP interface (a wide-area-network interface based on 3G data communication), but we emphasize that the analysis would be similar for the other TCP/IP-based network interfaces. In total, 100, 653 frames were captured on this interface during the experiment, 50, 296 IP packets were received by the gateway, and 41, 938 IP packets were sent by it. As a whole, the IP traffic exchanged with the gateway is composed of 93.8% of TCP packets, 4.1% of UDP packets and 2.1% of ICMP packets.

4.2. Attack detection results

In this section, we present the empirical results obtained for the detection of TCP SYN attacks. Using Scapy, we wrote a Python script for generating such attacks. The resulting pcap files can be used to train the learning algorithm, in addition to the "normal traffic" captured during the experiment described above. Moreover, using the utility tool

mergecap, it is possible to insert a SYN attack into the files containing the packets captured during the experiment, thereby allowing to test the learning algorithm.

As an example, Figure 3a plots the time-series for the difference between the numbers of initiated and established TCP connections per time slot (10s) which was extracted from a pcap file obtained using the above procedure. As can be observed in Figure 3b, the Dense RNN models correctly predicts that there was an attack.

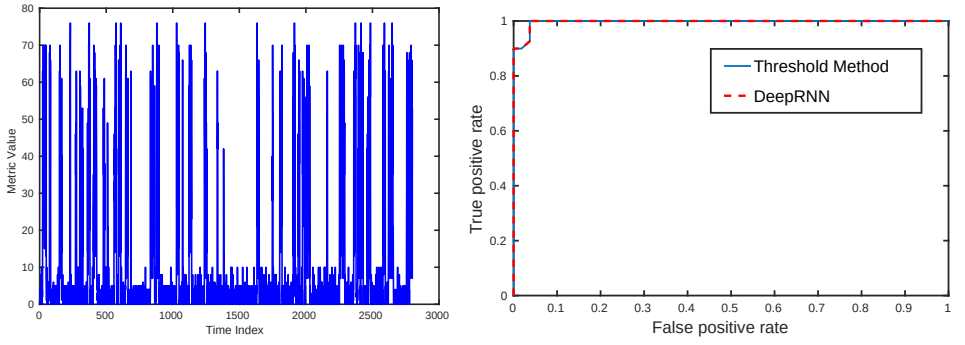


(a) (b)

Fig. 3: Scenario where a SYN attack, starting at 10:10 AM and lasting for 40 seconds, was inserted into the normal traffic captured from 9:15 AM to 11:03 AM on Sep. 21st, 2017: (a) time-series of the difference between the numbers of initiated and established TCP connexions per time slot (10 s), and (b) attack probability predicted by the Dense RNN.

4.3. Comparison with a Simple Threshold Detector

In order to compare the proposed approach with standard methods, we have implemented a simple threshold detector, and we report on the comparison of the DeepRNN approach against the threshold detector for the same attack data set. We have inserted multiple multiple SYN attacks, so that the diagram in Figure 8 (a) shows the same data as in Figure 7 (b) except for two aspects. Figure 8 (a) shows the full values of the data in 7 (a) (and not the differences), and furthermore it shows a large number of SYN attacks over time (the tall spikes). The Receiver Operating Curve in Blue in Figure 8 (b), gives the locus of the probability of attack (y-axis) versus the probability of false alarm (x-axis), as the detector threshold is varied from low values of the threshold (high values of the y-axis) to high threshold values (near the point [0,0]). The Red curve shows the effect of using the *trained* DeepRNN's performance, when it is tested with the thresholded data.



(a)

Fig. 4: On the left-hand-side we show the attack metric plotted against time. On the right-hand-side we show he Receiver-Operating-Curve, or locus of the probability of attack (y-axis) versus the probability of false alarms (x-axis) as the detection threshold is varied. We are comparing the DeepRNN attack detector (Red) with a standard threshold based detector (Blue).

5. Conclusion

In this paper, we have presented a methodology for the online detection of network attacks against IoT gateways. The methodology, which is based on a deep-learning approach with dense random neural networks, can predict the probability that a network attack is ongoing from a set of metrics extracted from packet captures. We have also observed that the results obtained in this manner are comparable to those obtained with a simple threshold detector.

In future work, we intend to apply our methodology to a broad range of network attacks, including Denial-of-Sleep attacks against Zigbee and Bluetooth-connected devices, and to investigate the design of a one-class classification algorithm for network attack detection.

Acknowledgements

This paper was supported by European Union's H2020 research and innovation programme under grant agreement No 740923, project GHOST[1] (Safe-Guarding Home IoT Environments with Personalised Real-time Risk Control).

References

[1] R. Rehim. Python Penetration Testing Cookbook Packt Publishing, 2017.
[2] E. Skoudis and T. Liston. Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses, 2nd Edition. Prentice Hall, 2005.
[3] M. Brownfield, Y. Gupta, and N. Davis. Wireless sensor network denial of sleep attack. In *Proc. 2005 IEEE workshop on information assurance and security*, United States Military Academy, West Point, NY, 2005.
[4] S. D. Dalrymple. Comparison of zigbee replay attacks using a universal software radio peripheral and usb radio. Master's thesis, AFIT, USAF, 2014.
[5] Alessio Di Mauro, Xenofon Fafoutis, Sebastian Mödersheim, and Nicola Dragoni. *Detecting and Preventing Beacon Replay Attacks in Receiver-Initiated MAC Protocols for Energy Efficient WSNs*, pp 1–16. Springer Berlin Heidelberg, 2013.
[6] A. Dubey, V. Jain, and A. Kumar. A survey in energy drain attacks and their countermeasures in wireless sensor networks. *Int. J. Eng. Res. Technol.*, 3(2), 2014.
[7] R. Falk and H-J. Hof. Fighting insomnia: A secure wake-up scheme for wireless sensor networks. In *3rd International Conference on Emerging Security Information, Systems and Technologies, 2009. IEEE SECURWARE'09.*, pages 191–196, 2009.
[8] F. Francois, O. H. Abdelrahman, and E. Gelenbe. Impact of signaling storms on energy consumption and latency of lte user equipment. In *2015 IEEE 7th Int. Symp. on Cyberspace Safety and Security*, pp 1248–1255, Aug 2015.
[9] Yee Wei Law, Marimuthu Palaniswami, Lodewijk Van Hoesel, Jeroen Doumen, Pieter Hartel, and Paul Havinga. Energy-efficient link-layer jamming attacks against wireless sensor network MAC protocols. *ACM Trans. Sen. Netw.*, 5(1):6:1–6:38, February 2009.
[10] X. Lu, M. Spear, K. Levitt, N. S. Matloff, and S. F. Wu. A synchronization attack and defense in energy-efficient listen-sleep slotted mac protocols. In *2008 2nd Int. Conf. on Emerging Security Information, Systems and Technologies*, 2008.
[11] M. Pirretti, S. Zhu, N. Vijaykrishnan, P. McDaniel, M. Kandemir, and R. Brooks. The sleep deprivation attack in sensor networks: Analysis and methods of defense. *Int. Journal of Distributed Sensor Networks*, 2(3):267–287, 2006.
[12] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. 7th Int. Workshop Security Protocols, Springer-Verlag, 1999.
[13] E. Y. Vasserman and N. Hopper. Vampire attacks: Draining life from wireless ad hoc sensor networks. *IEEE Trans. Mobile Computing*, 12(2):318–332, Feb 2013.
[14] E. Gelenbe and Y. Yin. Deep Learning with Dense Random Neural Networks. *Proc. Int. Conf. on Man–Machine Interactions*, Springer, 3–18, 2017.
[15] Y. Yin and E. Gelenbe. Single-cell based random neural network for deep learning. *Neural Networks (IJCNN), 2017 Int. Joint Conference on*, IEEE, 86–93, 2017.
[16] E. Gelenbe and Y. Yin. Deep learning with random neural networks. *Neural Networks (IJCNN), 2016 Int. Joint Conference on*, IEEE, 1633–1638, 2016.
[17] E. Gelenbe. Learning in the recurrent random neural network. *Neural Computation*, 5(1), 154–164, 1993.
[18] E. Gelenbe. Random neural networks with negative and positive signals and product form solution. *Neural Computation*, 1(4), 502–510, 1989.
[19] A. Collen, N. A. Nijdam, J. Augusto-Gonzalez, S. K. Katsikas, K. M. Giannoutakis, G. Spathoulas, E. Gelenbe, N. Ghavami, M. Volkamer, P. Haller, A. Sànchez and M. Dimas. GHOST - Safe-Guarding Home IoT Environments with Personalised Real-time Risk Control. in Proceedings of the 2018 ISCIS Security Workshop, Imperial College London, E. Gelenbe, P. Campegiani, T. Czachorski, S. Katsikas, I. Komnios, L. Romano, D. Tzovaras Eds., Lecture Notes in CCIS No. 821, Springer Verlag, Berlin, 2018.

[1] https://www.ghost-iot.eu