Random Neural Networks and Deep Learning for Attack Detection at the Edge

Olivier Brun
LAAS-CNRS, Université de Toulouse, CNRS
Toulouse, France

Yonghua Yin
Intelligent Systems and Networks Group
Electrical & Electronic Engineering Department
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.**

***Index Terms*—Cybersecurity, IoT, attack detection, deep learning, dense random neural network, Fog Computing.**

I. Introduction

With the proliferation of network attacks aiming at fraudulently accessing sensitive information 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 [1]. 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.

Our approach relies on a two-class classification algorithm based on a Dense Random Neural Network (RNN). This approach is an extension of the work presented in [2], [3].

The paper is organized as follows. In Sec. II, we analyze the vulnerabilities of IoT gateways and identify the relevant metrics for detecting attacks. We present our approach in Sec. III, whereas Sec. IV presents validation results. Some conclusion are drawn in Sec. V.

II. Vulnerabilities of IoT Environments

Figure 1 describes the architecture of an IoT environment, in which the main components are the smart devices. In a typical IoT environment, there may be dozens or even hundreds of sensors and actuators with various functions (e.g., measuring temperature, light, noise, etc.) Each of these devices may use different wireless technologies to communicate (e.g., Wi-Fi, Bluetooth, Ethernet, ZigBee and others). Due to tights limitations on hardware cost, memory use and power consumption, these wireless technologies have their own vulnerabilities, including traffic eavesdropping, packet replay, energy and collision attacks. A first possibility for an attacker is therefore to exploit the vulnerabilities of the various wireless sensor networks (WSN) present in an IoT environment.

Wireless sensor technologies used in IoT environments are not designed to enable smart devices to connect directly to the Cloud. Instead, they connect to an IoT gateway, which is a device capable of processing and aggregating sensor data, before sending it to the Cloud for further processing. Another possibility for an attacker is therefore to launch traditional TCP/IP attacks against IoT gateways.

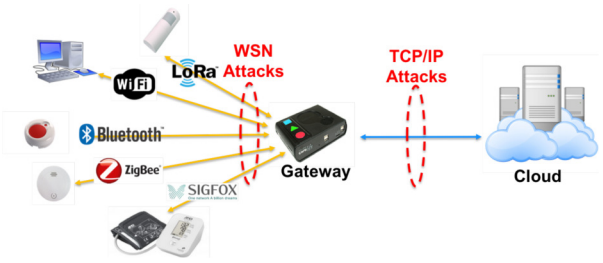


Fig. 1. Architecture and vulnerabilities of IoT environments.

Safeguarding IoT environments implies to be able to detect and block both types of attacks. The main difficulty here is that attackers have devised a myriad of different attack techniques. Our approach is to focus on some of the most common and most damaging ones:

1) Denial-of-Service attacks against TCP/IP networks: A denial-of-service attack (DoS attack) is typically accomplished by flooding the targeted host with superfluous requests in an attempt to overload systems and prevent some or all legitimate requests from being

fulfilled [4]. 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. Some DoS attacks remotely stop a network service on the victimized host by sending a malformed packet (e.g., Ping-of-death, Jolt2, Land, Latiera and Rose attacks). Another form of DoS attack aims at remotely exhausting the resources of the victim by flooding it with a huge number of packets (e.g., TCP SYN flood, UDP flood, ICMP flood attacks , HTTP POST DoS attacks, etc).

2) Denial-of-Sleep attacks against WSNs: A simple form of attack is to deplete the energy available to operate the wireless sensor nodes [5]–[7]. For instance, vampire attacks are routing-layer resource exhaustion attacks aiming at draining the whole life (energy) from network nodes, hence their name [8]. Denial-of-Sleep attacks are another form of energy attacks. These MAC-layer attacks aim at keeping the victim node out of its power conserving sleep mode, so as to reduce its lifetime. These attacks can take on several forms: sleep-deprivation attacks [9], barrage attacks, synchronization attacks [10], replay attacks [11], broadcast attacks [12] and collision attacks [13].

We have carefully analyzed the principles of the above mentioned attacks and identified the relevant metrics for detecting them. As an example, consider a UDP flood attack, in which the attacker sends a large number of UDP packets to random ports on a remote host. Upon reception of a UDP packet, the victim first checks for the application listening on this port, and, after seeing that no one listens on the port, it replies with an ICMP "Destination Unreachable" packet. In this way, the victimized system is forced to send many ICMP packets, eventually leading it to be unreachable by other clients, or even to go down. For this attack, the relevant metric is clearly the number of ICMP "destination unreachable" messages sent by the gateway over a time slot of given length. Indeed this metric is expected to take very low values under normal traffic conditions and to suddenly increase when a UDP flood attack is launched.

III. Attack Detection Approach

As shown in Fig. 2, 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" and attack IoT traffic, takes as input these metrics and predicts the probability that the IoT-connected home environment is currently under attack. Note that, although the computation of metrics is done on the gateway, the classification algorithm can be run in a nearby micro datacenter.

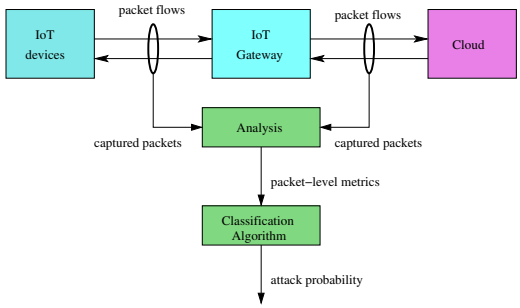


Fig. 2. Architecture of the online attack detection system..

We describe below the main components of our attack detection system.

A. Analysis of Packet Captures

The packet captured on the network interfaces of the IoT gateway are analyzed using Scapy, a packet manipulation tool for computer networks [14]. We have written a Python module based on Scapy. The module takes as input a pcap file and computes the values of various network metrics over a time slot (e.g., 10 seconds). Some metrics are rather generic (e.g., number of packets over a time slot) whereas others are devised to detect specific network attacks (e.g., number of ICMP "Destination unreachable" packets). By collecting the time-series for the different network metrics, we obtain a dataset which is used to train the classification algorithm, which in turn predicts the probability of an attack.

B. Network-attack detection with random neural network

We apply the random neural networks (RNN) [15], [16] developed for deep learning recently [17]–[19] to detecting network attacks using network metrics extracted from the captured packets, which can be viewed as a binary classification problem.

1) Dataset construction: We extract data samples from the time-series network metrics 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 inputs are network metrics and the output is a binary value.

2) Dense random neural network for deep learning: A dense cluster in a "Dense RNN" [17], [18] is composed of n statistically identical cells. Each cell receives inhibitory spike trains from external cells with rate x, whose spike behaviours follow the pattern of random selection of soma-to-soma interactions. A cell receives excitatory and inhibitory spikes from external world with rates λ^+ and λ^- respectively. Let p be the repeated-firing probability when a cell fires, r be the firing rate of a cell, and let q denote the probability of the activation state of a cell in the cluster in the steady state. Previous work shows that a numerical solution can be obtained for q such that

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

where $c = \lambda^+ p + rp - \lambda^- n - r - \lambda^+ pn - npr$ and $\phi(x) = (c - nx)^2 - 4p(n-1)(\lambda^- + x)n\lambda^+$. For notation ease, $\zeta(\cdot)$ is used as a term-by-term function for vectors and matrices.

Dense RNN in multi-layer architectures (DenseRNN) are 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:

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

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 [17], [18] have developed an efficient training procedure for DenseRNN to determine the values of $W_1, \cdots, W_L, W_{L+1}$, which combines unsupervised and supervised learning techniques [17]–[19].

IV. Experimental Results

A. 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 network interfaces (see Fig. 3).

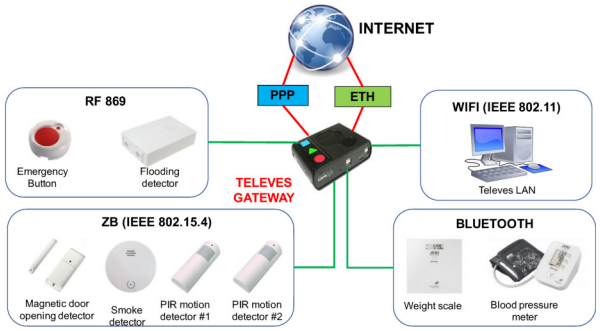


Fig. 3. Configuration used for the experiment.

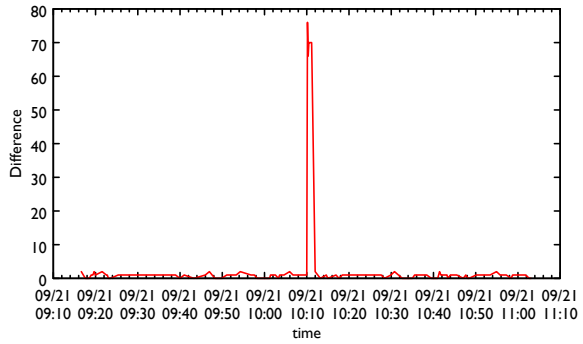
In the following, we focus on the packets captured on the PPP interface, but the analysis would be similar for the other network interfaces. In total, $100,653$ frames were captured on this interface during the experiment, $50,296$ IP packets were received by the gateway, and $41,938$ IP packets were sent by it.

B. Attack generation

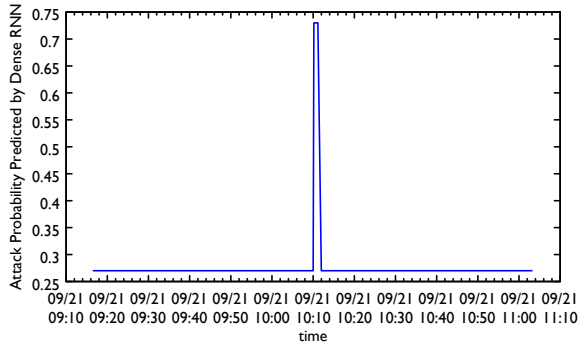
Using Scapy, we wrote a Python script implementing several attack techniques. The resulting pcap files can be merged with the initial packet captures using the utility tool *mergecap*, so as to superimpose an attack upon the "normal" traffic collected during the experiment.

C. Example of attack detection results

As an example, Fig. 4a plots the time-series for the difference between the numbers of initiated and established TCP connections per time slot (10 s) which was extracted from a pcap file obtained using the above procedure. As can be observed in Fig. 4b, the Dense RNN models correctly predicts that there was an attack.



(a)



(b)

Fig. 4. 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.

V. Conclusion

In this paper, we presented a methodology for the online detection of network attacks against IoT gateways using dense RNNs. As 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.

Acknowledgment

This paper was supported by European Union's Horizon 2020 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] K. Fu and T. Kohno and D. Lopresti and E. Mynatt and K. Nahrstedt and S. Patel and D., Richardson and B. Zorn B. Safety, Security, and Privacy Threats Posed by Accelerating Trends in the Internet of Things. http://cra.org/ccc/resources/ccc-led-whitepapers/, 2017.

[2] O. Brun and Y.Yin and J. Augusto-Gonzalez and M. Ramos and E. Gelenbe, IoT Attack Detection with Deep Learning, ISCIS Security Workshop, Imperial College, London, UK, Feb. 26 2018.

[3] O. Brun, Y. Yin and E. Gelenbe. Deep Learning with Dense Random Neural Network for Detecting Attacks against IoT-connected Home Environments. Procedia Computer Science, 134: 458-463, 2018.

[4] E. Skoudis and T. Liston. Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses, 2nd Edition. Prentice Hall, 2005.

[5] 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.

[6] 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.

[7] E. Gelenbe and Y. Murat Kadioglu. Energy life-time of wireless nodes with and without energy harvesting under network attacks. In *IEEE Int. Conf. on Communications (ICC)*, Kansas City, MO, USA, 20-24 May 2018.

[8] 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.

[9] 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.

[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] 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.

[12] M. Brownfield, Y. Gupta, and N. Davis. Wireless sensor network denial of sleep attack. In *Proc. 2005 IEEE workshop on information assurance and security*, West Point, NY, 2005.

[13] 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.

[14] R. Rehim. Python Penetration Testing Cookbook Packt Publishing, 2017.

[15] E. Gelenbe. Learning in the recurrent random neural network. *Neural Computation*, 5(1), 154–164, 1993.

[16] E. Gelenbe. Random neural networks with negative and positive signals and product form solution. *Neural Computation*, 1(4), 502–510, 1989.

[17] E. Gelenbe and Y. Yin. Deep learning with random neural networks. *Neural Networks (IJCNN), 2016 Int. Joint Conference on*, IEEE, 1633–1638, 2016.

[18] E. Gelenbe and Y. Yin. Deep Learning with Dense Random Neural Networks. *Proc. Int. Conf. on Man–Machine Interactions*, Springer, 3–18, 2017.

[19] 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.

[20] Y. Yin and E. Gelenbe. Nonnegative autoencoder with simplified random neural network. *CoRR*, abs/1609.08151, http://arxiv.org/abs/1609.08151, 2016.

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