IoT Attack Detection with Deep Learning

Olivier Brun, Yonghua Yin, Javier Augusto-Gonzalez, Manuel Ramos, Erol Gelenbe

IoT Attack Detection with Deep Learning

Olivier Brun[1,2], Yonghua Yin[1], Javier Augusto-Gonzalez[3], Manuel Ramos[3], and Erol Gelenbe[1]

[1] Imperial College London, London, UK,
[2] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[3] TELEVES, Madrid, Spain

Abstract. In this paper, we analyze the network attacks that can be launched against IoT gateways, identify the relevant metrics to detect them, and explain how they can be computed from packet captures. We also present the principles and design of a deep learning-based approach for the online detection of network attacks. Empirical validation results on packet captures in which attacks were inserted show that the Deep Neural Network correctly detects attacks.

Keywords: cybersecurity, IoT, attack detection, deep learning

1 Introduction

With the rise of the Internet of Things (IoT), the number and diversity of connected devices is expected to increase exponentially. While this promises significant benefits to users, who will have access to a broad range of new applications, it also opens the doors for a number of new security, privacy and safety threats, including physical safety and personal security threats. Tight limitations on hardware cost, memory use and power consumption of IoT devices have given rise to a number of security vulnerabilities[4], which can usually be dealt with traditional cyberattack countermeasures. However, as discussed in [23], protecting collections of smart devices poses new challenges that go far beyond securing each of the individual devices. These new challenges stem partly from the ability of smart devices to control physical aspects of their environment, and partly from their interactions with each other and with the Cloud. As a consequence, there is a wide consensus that security is one of the most challenging requirements for future IoT systems.

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. 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), in addition to actuators for controlling systems such as the heating, ventilation, and air conditioning system. Each of

[4] According to a 2014 HP study [22], 70% of IoT Devices are vulnerable to various attacks.

these devices may use different 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.

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.

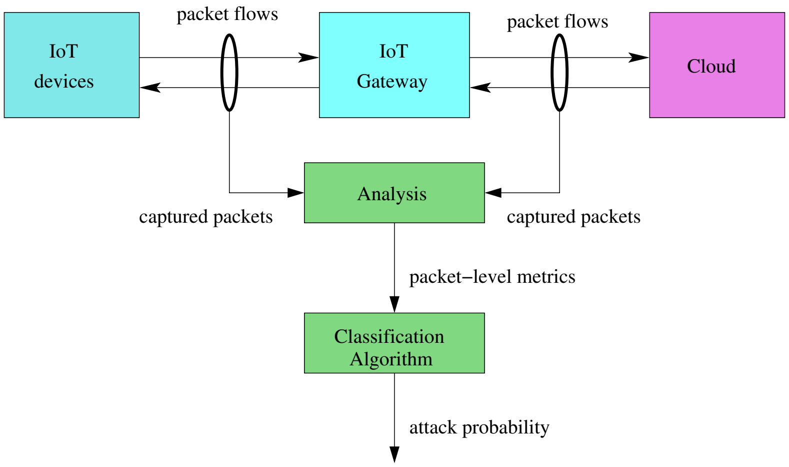


Fig. 1: Architecture of the online attack detection system..

The present paper is an extension of the work presented in [24]. Whereas a two-class classification algorithm based on a Dense Randon Neural Network was used in [24], implying that the learning algorithm needed to be first trained with attack traffic as well as ”normal” traffic, the present work proposes a one-class classification algorithm based on a Deep Neural Network which does not need to learn what is attack traffic.

The paper is organized as follows. In Sec. 2, we analyze the vulnerabilities of IoT gateways and identify the relevant metrics for detecting some of the attacks against them. Sec. 3 is devoted to the description of the deep learning algorithm. Section 4 presents the experiment setup as well as some preliminary results on flood attack detection. Some conclusions are then drawn in Section 5, where future research directions are also discussed.

2 Network Attacks

IoT gateways sit at the intersection of edge devices (sensors and actuators) and the Internet, and are therefore vulnerable to both traditional IP attacks targeted against the IoT gateway and to attacks against wireless smart devices. In this section, we focus on the security of IoT gateways and 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 for TCP/IP networks, and Denial-of-Sleep attacks for wireless sensor networks.

2.1 Denial-of-Service Attacks

A denial-of-service attack (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. 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 aim at remotely stopping a service on the victim host. The basic method for remotely stopping a service is to send a malformed packet. Below, are two standard examples of this type of attacks:

- **Ping-of-death attack** : the attacker tries to send an over-sized ping packet to the destination with the hope to bring down the destination system due to the system's lack of ability to handle huge ping packets.
- **Jolt2 attack** : the attacker sends a stream of packet fragments, none of which have a fragment offset of zero. The target host exhausts its processor capacity in trying to rebuild these bogus fragments.

Other well known examples of this type of attacks include *Land attacks*, *Latierra attacks* and *Rose attacks*, but there are many more.

Another form of DoS attack aims at remotely exhausting the resources of the victim host. This form of attacks involves flooding the remote victim with a huge number of packets. Below are some well-known examples:

- **TCP SYN attacks** : This type of attacks exploits a flaw in some implementations of the TCP three-way handshake. When an host receives the SYN request from another host, it must keep track of the partially opened connections in a "listening queue" for a given number of seconds. The attacker exploits the small size of the listen queue by sending multiple SYN requests to the victim, never replying to the sent back SYN-ACK. The victim's listening queue is quickly filled up, and it stops accepting new connections.
- **UDP flood** : The attacker sends a large number of UDP packets to random ports on a remote host. The victims checks for the application listening on this port. After seeing that no application 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.

There are of course many other forms of flooding attacks, including *ICMP floods* and *HTTP POST DoS attacks*, and many more.

2.2 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 [2], 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[5]. A simple form of attack is to deplete the energy available to operate the wireless sensor nodes [4, 6, 7]. For instance, vampire attacks are routing-layer resource exhaustion attacks aiming at draining the whole life (energy) from network nodes, hence their name [12]. In this section, we shall focus on another form of energy attacks, which are MAC-layer attacks known as Denial-of-Sleep attacks. Below are some examples of denial-of-sleep attacks:

- **Sleep Deprivation Attack**: the ability of sensor nodes to enter a low power sleep mode is very useful for extending network longevity. The attacker launches a sleep deprivation attack by interacting with the victim in a manner that appears to be legitimate; however, the purpose of the interactions is to keep the victim node out of its power conserving sleep mode, thereby dramatically reducing its lifetime [10, 11, 5].
- **Barrage Attack**: As in the sleep deprivation attack, the attacker seeks to keep the victim out of its sleep mode by sending seemingly legitimate requests. However, the requests are sent at a much higher rate and aim at making the victim performs energy intensive operations. Barrage attacks are more easily detected than sleep deprivation attacks, which are carried out solely through the use of seemingly innocent interactions.
- **Broadcast Attack**: malicious nodes can broadcast unauthenticated traffic and long messages which must be received by other nodes before being possibly discarded for lack of authentication [1]. Such attacks are hard to detect since they have no effect on system throughput, and nodes that receive them waste energy.

Other forms of denial-of-sleep attacks include *Synchronization attacks* [9], *Replay attacks* [3], and *Collision attacks* [8].

2.3 Relevant metrics to detect attacks

Tab. 1 presents the relevant metrics for detecting the attacks described above.

[5] For instance, these attacks can be conducted with KillerBee, a python-based framework for attacking ZigBee and other 802.15.4 networks

Attack	**Metric**
UDP flood	Number of destination UDP ports per second Number of outgoing ICMP ”destination unreachable” packets
TCP SYN	Difference between the numbers of initiated and established connexions
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

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

3 Network-attack detection with deep neural network

This section describes the use of deep neural networks [20, 21, 13, 16, 14, 15] to detect network attacks, which can be seen as a one-class classification problem. First, we show how to construct training datasets from captured packets without attack. Then, we describe the procedures used for deep neural networks to learn the data and solve the classification problem.

3.1 Dataset construction

Starting with the captured packets, statistical data (e.g., the rate) in time series can be obtained from captured packets in only the non-attack case. We extract samples from the time-series statistical data by setting a sliding window with length l. When a sample $X_n \in R^{l\times 1}$ is extracted, we assign the initial label of this sample denoted as $y_n = 0$. Then, we have a dataset $\{(X_n, y_n)|n = 1, \cdots, N\}$, where the input is the statistical data extracted and the output is a scalar.

3.2 Learning Procedures

Deep neural network The feed-forward architecture of multi layers allows a deep neural network (DNN) to extract high-level representations from input data [20, 21, 13, 16, 14, 15]. Suppose the DNN has an input layer, L hidden layers and an output layer, where the input and output layers have linear activation $y = x$ while hidden layers have nonlinear activation $y = \psi(x)$. The activation function $\psi(x)$ can be different types, such as the Logistic, Tanh and ReLU functions. Let O_l denote the output of the lth layer $l = 1, \cdots, L + 2$. Suppose there is a data vector X_n, a forward pass of X in the DNN can be described as:

$$\begin{aligned} O_1 &= X_n, \\ O_l &= \psi(O_{l-1}W_{l-1} + B_l), \text{ for } l = 2, \cdots, L+1, \\ O_{L+2} &= O_{L+1}W_{L+1} + B_{L+2}, \end{aligned}$$

where W_l is the connecting weight matrix between the lth and $(l + 1)$th layers and B_{l+1} is the bias matrix for the $(l + 1)$th layer with $l = 1, \cdots, L + 1$. These are the adjustable parameters of the DNN which need to be determined by the training procedure.

Training Phase Given X_n, the output of the DNN is $O_{L+2}(X_n)$, and the adjustable parameters in the DNN are W_l and B_{l+1} with $l = 1, \cdots, L+1$. Let c denote the center of the DNN outputs of all the training samples. The problem of training the DNN for classification becomes

$$\min_{W_l, B_{l+1}, l=1,\cdots,L+1} \sum_{n=1}^{N} (O_{L+2}(X_n) - c)^2. \tag{1}$$

Stochastic Gradient Descent (SGD) or its variants could be used for solving (1). Every a certain number of iterations, the center c is updated with the most recent DNN outputs using

$$c = \frac{\sum_{n=1}^{N} O_{L+2}(X_n)}{N}. \tag{2}$$

Testing Phase After the training phase, we can use the DNN for classification with the following three steps:

1. First, we retrieve the latest center c.
2. Second, for an input X_n, we have its DNN output $O_{L+2}(X_n)$. The distance between this output and the center c can be calculated by $d = |O_{L+2}(X_n) - c|$.
3. Third, the probability that the input X_n is an attack is calculated by

$$p = \frac{2}{1 + \exp(-10d)} - 1. \tag{3}$$

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).

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 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. As shown in Fig. 3, IP packets were sent to 158 distinct destination IP addresses, and received from $2,375$ distinct origin IP addresses.

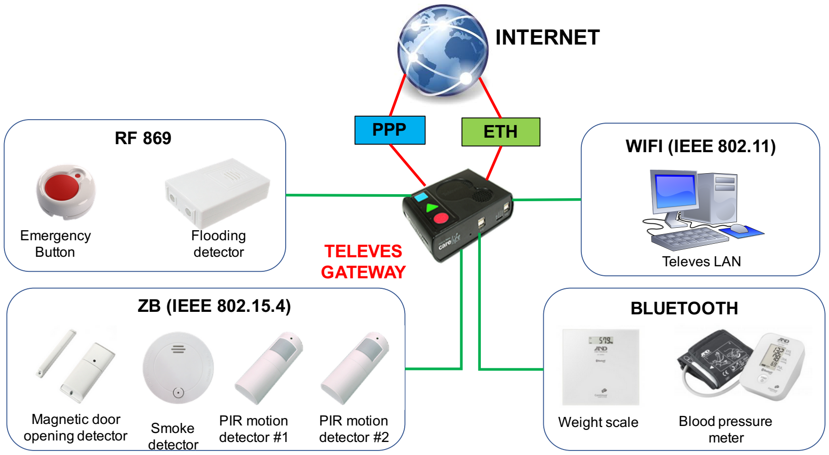
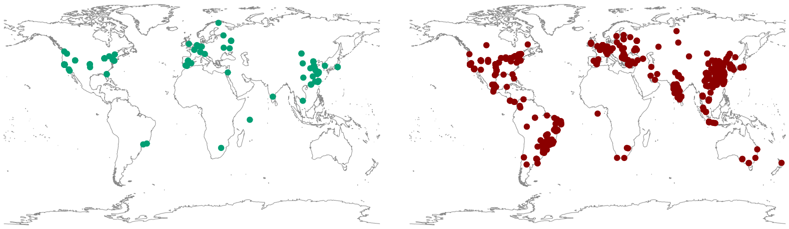


Fig. 2: Configuration used for the experiment.



(a) Packet destinations

(b) Packet origins

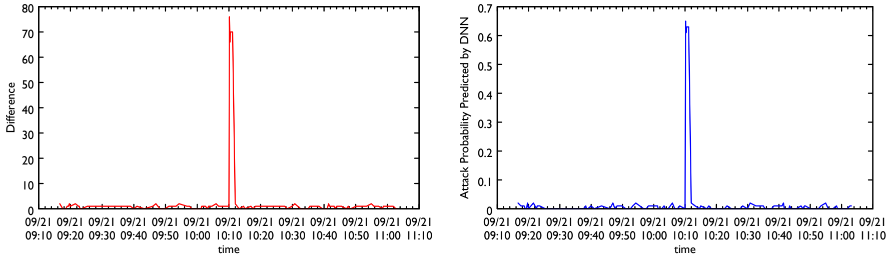
Fig. 3: Locations of packet origins and destinations.

The packet captured on the PPP interface were analyzed using Scapy. Scapy 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.

4.2 Attack detection results

TCP SYN Attacks We first 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 was merged with the initial packet capturesusing the utility tool *mergecap*, so as to superimpose a TCP SYN attack upon the "normal" traffic collected during the experiment.

For the detection of TCP SYN attacks, the relevant metric is the difference between the numbers of initiated and established TCP connections per time slot (10 s). This metric was extracted from initial packet captures in order to obtain training samples, where the length of the sliding window is set as 1. The number of training samples in the non-attack case is larger than that of the samples in attack case. The DNN exploited has two hidden layers, which respectively have 10 and 10 hidden neurons. The data in Fig. 4a is used to test the trained DNN. Fig. 4b plots the time-series for the probability predicted by the DNN that there is an attack. We can see that the DNN trained with only data in the non-attack case predicts correctly that there was an attack.



(a)

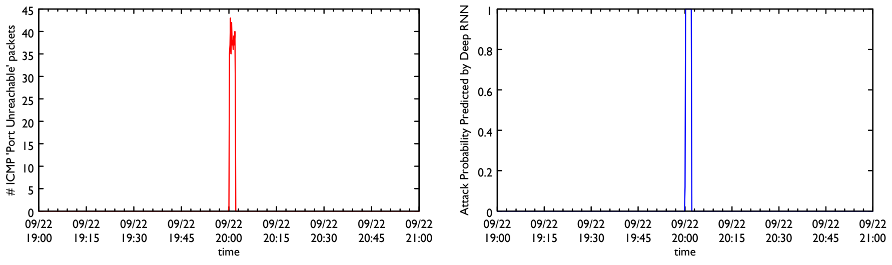
(b)

Fig. 4: Scenario where a SYN attack 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 classification algorithm.

UDP Flood Attacks We now consider UDP flood attacks. As for SYN flood attacks, we have used Scapy and mergecap to superimpose a UDP flood attack upon the "normal" traffic collected during the experiment. The attack was launched at 20:00 on September 22nd, 2017, and lasted for 2 minutes. For this type of attack, the relevant metric is the number of ICMP "port unreachable" messages sent by the gateway per time slot (10s). Again, this metric was extracted from initial packet captures in order to obtain training samples, which were feed to a DNN having similar characteristics to the one used for the detection of SYN attacks. The data in Fig. 5a is used to test the trained DNN. Fig. 5b plots the time-series for the probability predicted by the DNN that there is an attack. Again, it can be observed that the UDP flood attack is correctly detected.

5 Conclusion

In this paper, we presented a methodology for the online detection of network attacks against IoT gateways. The methodology, which is based on a deep-learning



(a) (b)

Fig. 5: Scenario where a UDP flood attack was inserted into the normal traffic captured on Sep. 22nd, 2017: (a) time-series of numbers of ICMP 'Port Unreachable' packets sent by the gateway per time slot (10 s), and (b) attack probability predicted by the classification algorithm.

approach, can predict the probability that a network attack is ongoing from a set of metrics extracted from packet captures. 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.

References

1. 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.
2. S. D. Dalrymple. Comparison of zigbee replay attacks using a universal software radio peripheral and usb radio. Master's thesis, AFIT, USAF, 2014.
3. 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.
4. 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.
5. 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.
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. 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.

9. 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.
10. 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.
11. F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. 7th Int. Workshop Security Protocols, Springer-Verlag, 1999.
12. 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.
13. E. Gelenbe and Y. Yin. Deep Learning with Dense Random Neural Networks. *Proc. Int. Conf. on Man–Machine Interactions*, Springer, 3–18, 2017.
14. Y. Yin and E. Gelenbe. Deep learning in multi-layer architectures of dense nuclei. *arXiv preprint arXiv:1609.07160*, 2016.
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. Y. Yin and E. Gelenbe. Nonnegative autoencoder with simplified random neural network. *CoRR*, abs/1609.08151, http://arxiv.org/abs/1609.08151, 2016.
20. Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks." Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. 2011.
21. Zhang M, Wu J, Lin H, Yuan P, Song Y. The Application of One-Class Classifier Based on CNN in Image Defect Detection. Procedia Computer Science. 2017 Dec 31;114:341-8.
22. HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack, `http://www8.hp.com/us/en/hp-news/press-release.html?id=1744676#.WmePCSOcbSw`, 29 07 2014.
23. 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.
24. O. Brun and Y.Yin and Y. Murat Kadioglu and E. Gelenbe. Deep Learning with Dense Random Neural Network for Detecting Attacks against IoT-connected Home Environments. submitted to 20th International Conference on Neural Networks (ICNN 2018).