Performance Evaluation of an Artificial Neural Network Multilayer Perceptron with Limited Weights for Detecting Denial of Service Attack on Internet of Things

Fernando M. de Almeida, Admilson de R. L. Ribeiro, Edward D. Moreno, Carlos A. E. Montesco
Department of Computing
Federal University of Sergipe – UFS
São Cristóvão, Brazil
email: fernando.m.al.91@gmail.com, admilson@ufs.br, edwdavid@gmail.com, estombelo@gmail.com

***Abstract*—One way to prevent attacks to security in the Internet of Things (IoT) is the adoption of an Intrusion Detection System (IDS). With the use of an Artificial Neural Network (ANN) it is possible to decrease the limitations of the IDS such as false positive that can compromise the system. In this paper, we evaluate the performance of two ANNs to verify which of both is the more adequate to use in an IDS for the IoT environment. We compare the performance of a Multilayer Perceptron (MLP) with Limited Weights with a Multilayer Perceptron with normal weights. The used Multilayer Perceptron presents ten neurons in the hidden layer. The implementation is in C language and run on an embedded platform with an ARM Cortex-M3 micro-controller. It is possible to consider the ANN training in another platform and to permit the embedded platform receives the trained weights. It is also possible to make the training in real time using the received data one time. We conclude that it is viable to use an Artificial Neural Network Multilayer Perceptron in an Intrusion Detection System for the Internet of Things.**

Keywords-IDS System; IoT Internet of Things; Multilayer Perceptron; Neural Network; Limited Weights;

I. INTRODUCTION

The Internet of Things (IoT) is a novel paradigm whose concept is based on the ubiquitous presence of objects, like sensors, actuators, mobile devices and Radio-Frequency Identification (RFID) tags. These objects can interact through single address to achieve common objectives [1].

Anytime, anywhere it will be possible to communicate with anything, a new dimension will be added to communication technologies [2]. The IoT can also be defined as a Wireless Sensor and Actuator Network (WSAN) connected to the Internet, and these sensors and actuators are the atomic components connecting the real world to the digital world [3]. The IoT is extremely vulnerable to attacks, most of the communication is wireless, most of the components have constrained resources and it is possible to physically attack the IoT components [1][4]. Considering that the IoT will have information about almost everything, security and privacy are key concerns in IoT research [5][6].

Xu, He and Li [5] say that the research about security is necessary to the massive adoption of this technology in the industry. Gubbi, Buyya, Marusic and Palaniswami [6] highlight the need of self-protection in domestic applications, arguing that actuators will be connected to the system and they will need protection from intruders.

The IoT, with its potential dimension and attack possibilities, needs a proper feature that allows it to keep safe with minimal human intervention otherwise its scalability will be compromised. According to Roman, Zhou and Lopez [7], fault tolerance will be essential in the IoT. The number of vulnerable systems and attacks will increase, so it is needed to develop intrusion detection and prevention systems to protect the components of the IoT.

The adoption of an Intrusion Detection System (IDS) allows the network to handle attacks. The design of IDS should evaluate the deployment environment. In the case of the Internet, it is important to have a high accuracy rate with a low false positive rate. In Wireless Sensor Network (WSN) should also consider the consumption of resources, such as energy and memory. In the IoT environment, as well as a WSN, there must be concern about the limited resources.

The use of artificial intelligence methods reduces the limitations of IDS [8], such as missing detections and false alarms that can compromise the system. The artificial intelligence method that we use in this work, Artificial Neural Networks (ANN), can be used to indicate the presence of an intruder, from environment features [9]. These features can be: communication duration, source address, destination address, and other information obtained from the environment.

The objective of this work is to implement a Multilayer Perceptron (MLP) optimized in memory and verify the feasibility of using it in IDS system for the IoT environment. The results of the accuracy and false positive rates were compared with related works of IDS that use ANNs. There is also the analysis of memory consumption of the implementation. The MLP was implemented with limited weights and without limited weights, to compare their memory consumption in IDS system using ANNs. The MLP is trained with Quantized Back-Propagation Step-by-Step and with an incremental method, where each input stimulates the ANN once to reduce the need of memory in the training phase.

This paper is organized in five Sections, as can be seen in the following: Section II shows related work, Section III presents the methodology of this work, Section IV shows and discusses the experiments and results and in the Section V there is the conclusion of this work.

 ISBN: 978-1-61208-473-2

II. RELATED WORK

The related works listed in this paper are split into two groups: IDS that uses ANNs and IDS for wireless resource constrained systems. This division was made because as far as we know, there is not an IDS for wireless resources constrained systems that use ANNs.

A. IDS that uses ANNs

The IDSs that uses ANNs described in this paper were not related to the resource constrained systems. So there is not any memory consumption evaluation making the comparison of memory consumption impossible.

Lei and Ghorbani [10] present an approach to an intrusion detection system with a neural network with a competitive learning approach. They achieve 4 times more performance in training than Self-Organizing Maps of Kohonen (SOM) with a better accuracy. Their proposal is called Improved Competitive Learning Network (ICLN). The winner neuron weight is updated with a value nearest to the entrance and the weights of the loser neurons are updated with values farthest to the entrance. The training used the KDD99 database, widely used in IDSs.

Eskin, Arnold, Prerau, Portnoy and Stolfo [11] provide a framework for detection of non-labeled anomalies. They perform tests alternating the use of Kernel function in the feature map and three algorithms: Cluster-based estimation, KNN and One Class SVM. Using the KDD99 database, they achieved a detection rate between 91% and 98% with a false positive rate between 8% and 10%.

Amini, Jalili and Shahriari [12] present two solutions related to IDS in unsupervised network. The experiments are made with three types of ANNs: ART-1, ART-2 and SOM. They achieve an accuracy rate between 95.74% and 97.42%, and false positive rate between 1.99% and 3.50%. They also use the KDD99 database.

Yan, Wang and Liu [13] propose a hybrid technique that uses a rule-based decision and an ANN. When rules detect an abnormality, the packet is forwarded to a multilayer perceptron with 50 neurons in the hidden layer and other rule-based decision is made to detect the intrusion or not. Using the KDD99 database, they get an accuracy rate of 99.75% and a false positive rate of 0.57%.

B. IDS for Wireless Resource Constrained Systems

Raza, Wallgren and Voigt [14] designed, implemented and evaluated SVELT, an IDS for the IoT. The SVELTE detects sinkhole and selective-forwarding attacks in 6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks) wireless network that uses the RPL routing protocol. The IDS builds the network topology of RPL in the border router and uses algorithms to detect inconsistencies, possible filtered nodes, network topology validity and end-to-end losses.

Salmon *et al.* [15] proposed an anomaly based IDS for WSNs using the Dendritic Cell Algorithm (DCA). The proposed IDS architecture has five elements, distributed between roles in the network. The authors proposed two roles: Dendritic Cells (sensor-dc), responsible for sense the environment's values, managing the monitoring and parameter base, organize the tasks and coordinate the responses and actions to other managers; and the Lymph node (sensor-lymph) responsible to execute the dendritic cell algorithm, detect an attacker, manage the base rules and execute the actions to combat the identified attacks. Several experiments were made, changing configuration, time of jamming attack, number of sensor-dc. Through the tests, the authors concluded that the IDS proposed is efficient for WSNs saving energy from the nodes while there is a jamming attacker.

C. Related work analysis

All the related work listed in this paper that uses ANNs, uses the KDD99 database [10][11][12][13]. This database provides a big number of labeled connections that helps supervised training, although it is used in unsupervised training [11][12].

In supervised training, the accuracy rate is bigger than 95% [10][13], and have a small false positive rate. Using unsupervised training, it is possible to achieve an accuracy bigger than 90% with a false positive rate around 9% [11] or decrease the accuracy around 75% with a false positive rate between 2% and 3.5% [12].

The related works of IDS for wireless resource constrained systems do not use the KDD99 database. In Salmon *et al.* [15] work, the network topology focuses on jamming attack, which it is not characterized in the KDD99 database. In SVELTE IDS [14] the attacks are sinkhole and selective forwarding, also they are not characterized in the KDD99 database.

III. METHODOLOGY

The ANN that we chose is the MLP with ten neurons in the hidden layer. The choice of ten neurons in the hidden layer was made to minimize the memory consumption. The evaluation of the ANNs was made from a C language implementation, for the MLP and Multilayer Perceptron with Limited Weights (MLPLW). The MLP and MLPLW share the same interface, so the utilization is made with the same steps. The difference between the MLP and MLPLW implementation is in the structure that keeps the trained data and the training algorithm. In the classification phase, there is only need to adapt the value of each weight to represent a float point value. The MLP training algorithm is the Back-Propagation and the MLPLW training algorithm is the QBPSS [16].

The database used was KDD99, which is used in several papers [10][11][12][13] to validate an IDS. This database has connections from the transport layer, like UDP and TCP. Each connection is classified between a normal classification or some kind of attack. The attacks are grouped into four large groups, they are: Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L) and Probe. The database has 4,898,431 data items, each having 41 features.

Unfortunately, the KDD99 database cannot represent an IoT environment. The utilization of KDD99 database is important to check our ANN implementations and see if they can achieve similar results to related works. When the ANN implementation are validated, we can analyze the memory consumption and check if it can be used in an IoT environment.

For this work, it is considered the DoS attacks that can affect 6LoWPAN networks too, making IoT networks that

 ISBN: 978-1-61208-473-2

use this protocol stack vulnerable to DoS attacks using ICMP (Internet Control Message Protocol), UDP (User Datagram Protocol or TCP (Transmission Control Protocol).

The training of both networks, MLP and MLPLW, was performed on a PC, called traditional platform, whose features can be seen in Table 1. In a real environment of the IoT, this training can be performed in the cloud, for example, so the metrics of training are not crucial in this work. The third approach of training is using the MLPLW ANN, but performing the training in the constrained resource platform. The training is similar to Back-Propagation, but each input is trained once, considering that the platform will receive numerous packets and does not need to retrain each input, this training removes the need to keep training inputs in the platform, reducing the memory usage.

The first step is to normalize the KDD99 database. The discrete features are quantized, and each possible value represents a number. All the features, discrete or continuous, will have a value between 0 and 1, at the end of the normalization. The sigmoid function was chosen to be the activation function because it has its values between 0 and 1, like the normalized features.

The second step, after the normalization, is to train the ANN. Each input to the database will have an output from the ANN. The output is compared to the desired output and the error is calculated. In (1), it is possible to see the equation to calculate the error E. The desired output is represented by d, the input weight set is represented by $w^{(1)}$, and the hidden weight set is represented by $w^{(2)}$.

$$E = \sum_{j=0}^{n_1} w_j^{(2)} \cdot S\left(\sum_{k=0}^{n} x_k \cdot w_{kj}^{(1)}\right) - d \quad (1)$$

The neuron weights of the hidden layer are updated by a fraction of the error multiplied by each input feature. In (2) it is possible to see how the hidden weight set is updated, where the hidden weight update is represented by $\Delta w^{(2)}$, the learning rate of hidden layer is represented by η_2, the error is represented by E, and the output set of hidden layer is represented by Y. Equation (3) shows how the input weight set is updated, where the input weight update is represented by $\Delta w^{(1)}$, the output set of the input layer is represented by Y, the hidden weight set is represented by $w^{(2)}$, the learning rate of input layer is represented by η_1, the error is represented by E, and the input is represented by x.

$$\Delta w^{(2)} = -\eta_2 \cdot E \cdot Y \quad (2)$$

$$\Delta w^{(1)} = -\left(1 - Y^2\right) \cdot w^{(2)} \cdot \eta_1 \cdot E \cdot x \quad (3)$$

Each step of training consists in the update of hidden weights and input weights for an input. The set of steps to train the ANN by each input is called an epoch. The training is finished if the quadratic error, when used a testing set of data, has increased after ten epochs or if it reaches an arbitrary epoch value defined by the designer.

The MLPLW training is similar to the MLP, but after each step, the weight sets are quantized and the previous weight update is added to current weight update. These differences are made in the QBPSS (Quantized Back-Propagation Step-by-Step) [16] to speed up the training phase for a limited weight ANN.

The QBPSS training method is proposed by Bao, Chen and Yu [16] for ANNs with limited weights, reducing training time by up to 7 times. The QBPSS is inspired by Back-Propagation. The algorithm is similar to Back-Propagation, but it is considered a value proportional to previous adjust weight value, also the weight is quantized each step, in the beginning with a soft quantization until reach the quantized value expected.

Equation (4) shows the update of hidden weight set, where the hidden weight update is represented by $\Delta w^{(2)}$, the momentum for the height update is represented by δ, the learning rate for hidden layer is represented by η_2, the error is represented by E, the output set of the input layer is represented by Y, and the last hidden weight update is represented by $\Delta w_{i-1}^{(2)}$. Equation (5) shows the update of input weight set, where the input weight update is represented by $\Delta w^{(1)}$, the momentum for the weight update is represented by δ, the output set of the input layer is represented by Y, the hidden weight set is represented by $w^{(2)}$, the learning rate for the input layer is represented by η_1, the error is represented by E, the input is represented by x, and the last input weight update is represented by $\Delta w_{i-1}^{(1)}$.

$$\Delta w^{(2)} = -(1-\delta) \cdot \eta_2 \cdot E \cdot Y + \delta \cdot \Delta w_{i-1}^{(2)} \quad (4)$$

$$\Delta w^{(1)} = -(1-\delta) \cdot \left(1 - Y^2\right) \cdot w^{(2)} \cdot \eta_1 \cdot E \cdot x + \delta \cdot \Delta w_{i-1}^{(1)} \quad (5)$$

As the MLP training, the MLPLW training will continue to use all inputs to train the ANN until the quadratic error increases after 10 epochs or an arbitrary value defined by the designer is reached.

The second MLPLW training uses the methods and equations of the QBPSS, but each input stimulates the ANN once. The update of the weights is equal to the first MLPLW training method.

After the training, for both ANNs, the input dataset is used to verify the correct classification and the incorrect classification of each ANN, MLP and MLPLW. With this

information it is possible to calculate the accuracy rate and the false positive rate.

IV. EXPERIMENTS AND RESULTS

To perform the test of the two chosen ANNs, two platforms have been selected. One is a personal computer, called traditional platform, and the other is an embedded platform with an ARM Cortex-M3 micro-controller. Both technical features can be seen in Table I.

The choice of Arm Cortex-M3 processor was taken by the highlight of ARM core micro-controllers. The Contiki community, for example, is already adopting the ARM micro-controllers as the focus for the 3.0 version of the operating system [17].

TABLE I. CHOSEN TECHNICAL PLATFORM FEATURES

Technical Feature	Traditional platform	Embedded Platform
Processor	Intel I7-2630 QM	STM32F103VET6 (ARM Cortex-M3 core)
Processor Frequency	2.00 GHZ	72 MHz
Volatile Memory	8 GB	64 KB
Persistent memory	1 TB	512 KB

The measures made for both platforms fit the context of resource-constrained devices: ROM and RAM memory. There is also a measure to prove the efficiency of the ANN: Accuracy and false positive rate. For the incremental training, where each input is used once, it measures the accuracy and false positive rate for each thousand inputs to see the evolution of these measures according to the inputs.

The chosen ANN, MLP or MLPLW, was made as a first test to verify if this ANN configuration can be used in resource-constrained devices. Also, we selected to use ten neurons in the hidden layer to achieve less memory consumption with a greater accuracy rate. The RAM memory consumed by the hidden layer grows linearly proportional to the number of system inputs.

The training phase of both ANN used 90% of KDD99 data as training set and the remaining 10% to test, as the testing set. We used the same proportions of each type of connection in training and testing set to avoid vicious training of testing. After performing the training phase with KDD99 database in MLP and MLPLW ANNs, the trained networks are stimulated with the same input data. The predicted output is compared to the desired output and, if it is equal to the database label, the input is marked as correct, otherwise is marked as incorrect. If a normal connection is classified as an attack, it is marked as a false positive. The sum of correct inputs is compared to the total inputs and is possible to calculate the accuracy rate of the ANN. The sum of false positive inputs is compared to the total inputs and it is possible to calculate the false positive rate of the ANN. In the Table II, it is possible to see the results of the MLP and MLPLW. It is possible to see that the accuracy rate and false positive rate of the MLPLW remained close to the MLP. While average accuracy rate of MLP was 97,75% and the average false positive rate was 2,13%, the average accuracy rate of MLPLW was 97,65%, 0,10% lower than the MLP rate, and the average false positive rate was 2,11%, 0,02% higher than the MLP rate.

The results in Table II, were compared with the results of related works. This comparison is shown in the Table III, the bold lines are the ANNs of this paper. When compared with techniques of unsupervised ANNs, MLP and MLPLW achieve better results, but it should be considered that MLP and MLPLW use labeled data, that help the classification. When compared with the ANNs presented in [10] and [13], the MLP and MLPLW with ten neurons in the hidden layer have similar results.

TABLE II. ACCURACY AND FALSE POSITIVE RATE OF MULTILAYER PERCEPTRON AND MULTILAYER PERCEPTRON WITH LIMITED WEIGHTS.

ANN	Measure	Average	Standard Deviation
MLP	Accuracy rate	97,75%	0.02
	False Positive rate	2,13%	0.02
MLPLW	Accuracy rate	97,65%	0.01
	False Positive rate	2,11%	0.01

TABLE III. COMPARISON OF RESULTS OF THIS PAPER WITH RESULTS OF RELATED WORKS.

ANN	Accuracy	False Positive	Supervised training?
MLP	**97,75%**	**2,13%**	**Yes**
MLPLW	**97,65%**	**2,11%**	**Yes**
ICLN [10]	97.89%	–	Yes
Cluster [11]	93%	10%	No
K-NN [11]	91%	8%	No
SVM [11]	98%	10%	No
RT-UNNID ART-1 [12]	71.17%	1.99%	No
RT-UNNI ART-2 [12]	73.18%	2.30%	No
RT-UNNID SOM [12]	83.44%	3.50%	No
YAN; WANG; LIU [13] (50 neurons in hidden layer)	99.75%	0.57%	Yes

With the results of the MLP and MLPLW training, checking that they have similar results to other related works, both ANNs were trained with a different approach. Each input stimulated the ANN once and, we measured the accuracy after one thousand stimulations. Considering that in the IoT environment the nodes will receive several packets, the ANN can be trained while the node is alive. This experiment checks how fast the ANN can respond to a new type of DoS Attack.

The MLP achieves an accuracy rate of 97% after the seventh thousand connections, considering normal and attack connections, after that the accuracy rate is established around 97,4%. This curve is shown in the Figure 1.

 ISBN: 978-1-61208-473-2

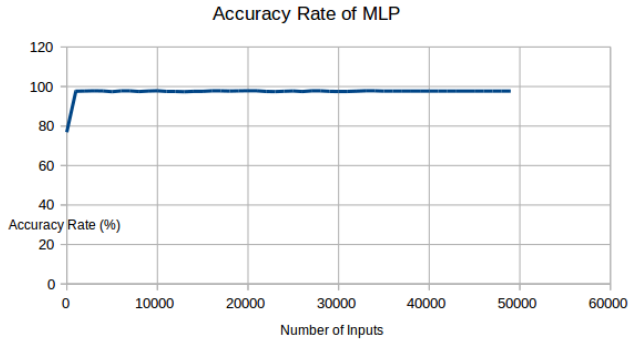

Figure 1. Accuracy rate of MLP after training with each input once

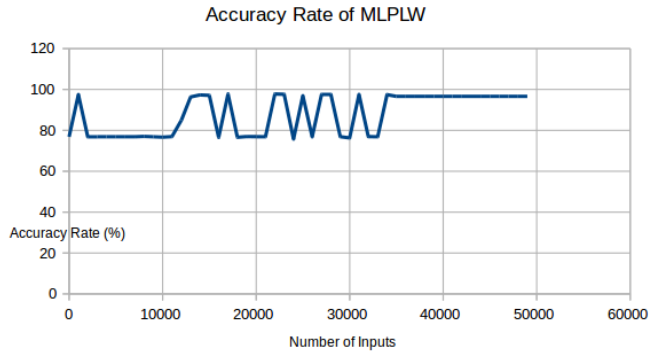

Figure 2. Accuracy rate of MLPLW after training with each input once

In the MLPLW ANN, the accuracy rate achieves 97% in the first thousand connections, but oscillates until the third four thousand connection. This curve is shown in the Figure 2. It is important to highlight that the input data is used in random order, using normal and attack data randomly. We can use a random order input because the KDD99 database is based on connections, and they do not need to be used in a specific order. The part of KDD99 database used has 76% of normal data and 24% of DoS attack data.

The measures related to resources used can be seen in Table IV. It is possible to see that MLPLW consumes more ROM memory than MLP, this is because of some procedures to transform the limited weight into the floating point weight, but when compared to ROM memory available in embedded platform, it is an increase of 0.03%. Besides the RAM memory consumption is smaller in MLPLW and, when compared with RAM memory, available in the embedded platform, it is a decrease of 4.5%. It is possible to see that the MLPLW has a smaller impact in the embedded platform memory.

The ROM memory consumed by MLPLW with training is the biggest, because it needs the code of QBPSS, but analyzing in percentage, the amount of ROM memory consumed is less than 0.5%. The RAM memory consumed is equal to MLPLW and the analysis is the same. It is important to highlight that the MLPLW with training do not need to communicate with a server, reducing the communication and energy consumption, so in the nodes where the energy is a crucial factor, the MLPLW with training presents this advantage.

TABLE IV. MEMORY USED BY MLP AND MLPLW NEURAL NETWORKS WITH REMOTE TRAINING

Resource	MLP	MLPLW	MLPLW with training
ROM memory	214 bytes	354 bytes	1716 bytes
RAM memory	3360 bytes	420 bytes	420 bytes
Related ROM (Embedded Platform)	0.04%	0.07%	0,33%
Related RAM (Embedded Platform)	5.12%	0.64%	0,64%

V. CONCLUSION

In this paper, we presented the performance evaluation of a Multilayer Perceptron with Limited Weights comparing with a Multilayer Perceptron with normal weights. The accuracy and false positive rate decreases 0,10% when the weights are limited to one byte. The ROM memory consumption increases 140 bytes for the weight limitation and 1362 bytes including the training algorithm. The RAM memory, when limiting the weights, decreases eight times.

With this experiment, it was possible to observe the possibility to use a multilayer perceptron in an embedded platform. The consideration of the training in another platform allows the embedded platform in this work to use an ANN trained by 4 million data, with more than 97% of accuracy and using only 354 bytes of ROM and 420 bytes of RAM, less than a kilobyte of memory.

When the ANNs are trained while the platform is running, there is a good response from the MLP, achieving a great accuracy rate for the DoS attacks after the seventh thousand connections. The MLPLW training should be revised to approximate it results to the MLP ANN.

With these results it is possible to achieve better use of ANNs in embedded systems connected to the Internet, using the techniques from the IDSs for the Internet, with high detection rate and low false positive rate, in resource-constrained platforms.

In a future work, it is intended to improve the MLPLW in-node training approximating of MLP in-node training. Also, implement the MLPLW in a 6LoWPAN environment and simulate it with live data.

ACKNOWLEDGMENT

This work was supported by CAPES, CNPq and FAPITEC/SE

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey”. Computer Networks, v. 54, n. 15, p. 2787-2805, 2010.

 ISBN: 978-1-61208-473-2

[2] L. Tan and N. Wang, “Future internet: The internet of things”. In: Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on. IEEE, 2010. p. V5-376-V5-380.

[3] M. Presser and A. Gluhak, “The internet of things: connecting the real world with the digital world”, EURESCOM mess@ge – The Maganize for Telecom Insiders, vol. 2, 2009. Available: http://archive.eurescom.eu/message/messageSep2009/The-Internet-of-Thing%20-Connecting-the-real-world-with-the-digital-world.asp. Retrieved: December, 2015.

[4] Q. M. Ashraf and M. H. Habaebi, “Autonomic schemes for threat mitigation in Internet of Things.” Journal of Network and Computer Applications 49, 2015, pp. 112-127.

[5] L. D. Xu, W. He, and S. Li, “Internet of things in industries: A survey.” Industrial Informatics, IEEE Transactions on 10.4, 2014, pp. 2233-2243.

[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions.” Future Generation Computer Systems 29.7, 2013, pp. 1645-1660.

[7] R. Roman, J. Zhou, and J. Lopez, “On the features and challenges of security and privacy in distributed internet of things.” Computer Networks 57.10, 2013, 2266-2279.

[8] D. D. Oliveira, R. J. P. B. Salgueiro, and E. D. Moreno, “A Danger Theory Immune-inspired Architecture for the Prediction of Security Attacks in Autonomic Networks”. In: Communications Workshop, 2013, Santiago, Chile. Proceedings of 5th IEEE Latin-american Conference on Communications, IEEE LATINCOM, 2013.

[9] H. H. Soliman, N. A. Hikal, and N. A. Sakr, “A comparative performance evaluation of intrusion detection techniques for hierarchical wireless sensor networks”. Egyptian Informatics Journal, v. 13, n. 3, 2012, pp. 225-238.

[10] J. Z. Lei and A. Ghorbani, “Network intrusion detection using an improved competitive learning neural network”. In: Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on. IEEE, 2004, pp. 190-197.

[11] E. Eskin, A. Arnold, and M. Prerau, L. Portnoy, S. Stolfo, “A geometric framework for unsupervised anomaly detection”. In: Applications of data mining in computer security. Springer US, 2002, pp. 77-101.

[12] M. Amini, R. Jalili, and H. R. Shahriari, “RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks”. Computers & Security, v. 25, n. 6, 2006, pp. 459-468.

[13] K. Q. Yan, S. C. Wang, and C. W. Liu, “A hybrid intrusion detection system of cluster-based wireless sensor networks”. In: Proceedings of the International MultiConference of Engineers and Computer Scientists, 2009, pp. 18-20.

[14] S. Raza, L. Wallgren, and T. Voigt, “SVELTE: Real-time intrusion detection in the Internet of Things.” Ad hoc networks 11.8, 2013, pp. 2661-2674.

[15] H. M. Salmon, et al. “Intrusion detection system for wireless sensor networks using danger theory immune-inspired techniques.” International journal of wireless information networks 20.1, 2013, pp. 39-66.

[16] J. Bao, Y. Chen and J. Yu, “An optimized discrete neural network in embedded systems for road recognition”. Engineering Applications of Artificial Intelligence, v. 25, n. 4, 2012, pp. 775-782.

[17] Contiki, “Contiki 3.x Roadmap”, unpublished. Available: https://github.com/contiki-os/contiki/issues/422. Retrieved: December, 2015.

 ISBN: 978-1-61208-473-2