A Self Organizing Map Intrusion Detection System for RPL Protocol Attacks

*Elie Kfoury, *Julien Saab, **Paul Younes, **Roger Achkar
*Department of Computer Science
**Department of Computer and Communications Engineering
American University of Science and Technology, Beirut
ekfoury@aust.edu.lb, {jps00001,pwy00001}@students.aust.edu.lb, rachkar@aust.edu.lb

***Abstract*—Routing over Low Power and Lossy Networks (RPL) is a standardized routing protocol for constrained Wireless Sensor Network (WSN) environments. The main node's constraints include processing capability, power, memory, and energy. RPL defines how the nodes in a WSN form a Mesh topology enabling them to route messages. Unfortunately, various attacks exist on the RPL protocol that can disrupt the topology and consume nodes' energy. In this paper, we propose an Intrusion Detection System (IDS) based on Self-Organizing Map (SOM) Neural Network to cluster the WSN routing attacks, and hence notify the system administrator at an early stage, reducing the risk of interrupting the network and consume nodes power. Results showed that the SOM is able to cluster routing packets with three different types of attacks, as well as clean data.**

Keywords*—*RPL, WSN, IDS, SOM, Neural Network, Energy.

I. Introduction

The Internet of Things (IoT) is a network of interconnected computing devices embedded in everyday objects, enabling them to exchange data in a Machine-to-Machine approach. This technology is empowering new services and business opportunities as the number of diverse IoT applications (smart homes, smart cities, smart vehicles [1], e-health and personal care, smart agriculture and others ...) is ultimately increasing. According to Gartner, the deployment of IoT devices is expected to reach almost $3 trillion in 2020. Hence, it is considered as a vital topic nowadays because of its influence different aspects of human lifestyle. A research conducted by Vanson Bourne indicates that security is one of the top three barriers preventing the adoption of IoT [2]. Results of a recent survey conducted by Altman Vilandrie & Company showed that almost half of all U.S companies that use IoT devices have been suffering from a security breach [3]. Therefore, securing IoT devices is important to foster their adoption by businesses and end costumers. Multiple solutions for securing IoT devices exist [4][5] , but most of them tackle the integrity and authentication problems, but not the availability.

An IoT network architecture is maintained based on various protocols (Fig. 1). For instance, a typical WSN (Wireless Sensor Network) include IEEE 802.15.4 on the physical and media access layers, IPv6, 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks), and RPL (Routing Protocol for Low power and Lossy Network) on the network layer, MQTT (Message Queuing Telemetry Transport) or CoAP (Constrained Application Protocol) on the application layer.

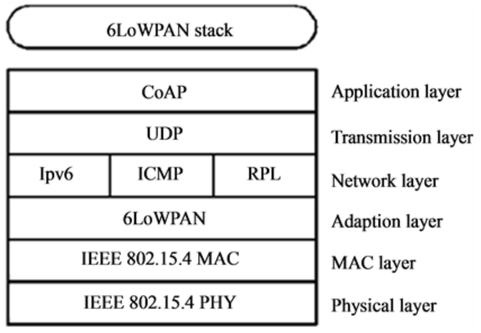


Fig. 1: 6LoWPAN Protocol Stack (OSI Reference)

RPL is a novel distance vector routing protocol standardized for constrained 6LoWPAN networks enabling nodes to communicate in a mesh topology. Unfortunately, several attacks exist on the RPL protocol that target a node's availability, and increase dramatically its power consumption.

In this paper we propose a novel Intrusion Detection System (IDS) for classifying RPL well-known attacks. Several attempts to create an IDS for RPL attacks have been introduced. Raza et al. [6] proposed SVELTE, Kaplantzis et al. [7] used Support Vector Machines (SVM), Livani et al. [8] relied on feature vectors. In our IDS, we used the Self-Organizing Map (SOM) neural networks to perform RPL attacks classification. Neural networks in general can be divided into a variety of types, and can be applied to different types of applications [9][10][11][12].

The main contributions of our IDS include: 1) The ability to detect multiple types of RPL attacks using unsupervised learning, 2) Enhancing power consumption by notifying the network administrator at an early stage about a certain attack, 3) Ensuring network availability due to the immediate notice of a security breach.

The paper is divided as follows: Section II describes some background on RPL protocol and how the WSN topology is formed. Next, it explains the RPL well-known attacks and the control messages frequency on each attack. Moreover, it explains the SOM and how it can build a *map* from input samples. Section III introduces the proposed system, Section IV discusses the simulation and results. Finally, we conclude with the intended future work.

II. Background and Literature Review

In this section, we give an overview on RPL as a routing protocol, its topology formation, well-known attacks, and their implications on node's power consumption. Furthermore, we elaborate on how SOM neural networks work.

A. *Routing Protocol for Low power and Lossy Network (RPL)*

Low-Power and Lossy Networks (LLNs) are a class of network in which both the routers and their interconnect are constrained [13]. RPL is a distance vector routing protocol that makes use of IPv6 in an LLN WSN network. The protocol tries to avoid loops by computing a node's position relative to other nodes within a destination oriented directed acyclic graph (DODAG). This DAG is formed after exchanging RPL control messages between the nodes. Fig. 2 illustrates the sequence of control messages exchanged between nodes. RPL supports unidirectional traffic towards the root and bidirectional among participating nodes.

1) *DODAG Information Object (DIO)*

Message multicasted downwards by a node in the DODAG to let other nodes know about it. This message is considered as an announcement to inform other nodes to join the DAG if they are interested.

2) *DODAG Information Solicitation (DIS)*

When no announcement is received, a DIS control message is sent by the node to search if any DODAGs exist. This is considered similar to the Router Solicitation (RS) in IPv6 Neighbor Discovery.

3) *DODAG Advertisement Object (DAO)*

A message sent by a child to a parent or root to allow the child to join the previously built DODAG. A DAO message might be acknowledged by a Destination Advertisement Acknowledgment (DAO-ACK) message back to the originator.

4) *DAO-ACK*

A response sent by a parent or root after a DAO message that permits or denies a child to join the DODAG.

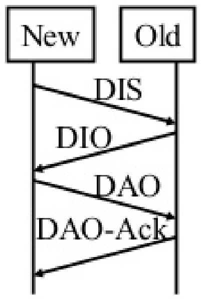


Fig. 2: RPL Control Messages Sequence Diagram (src: Washington University in St. Louis)

B. *Attacks against RPL*

In this section, we investigate the well-known security attacks against RPL in a WSN, the number of control messages per each attack, and their effects on the power consumption of a participating node.

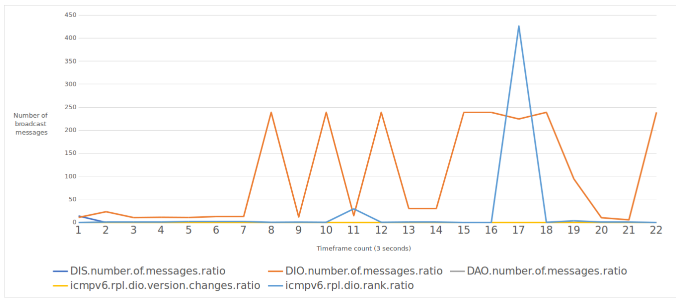


Fig. 3: RPL Control Messages Frequencies on a clean network

The number of Control messages in a clean network behavior is depicted in Fig. 3. It is obvious that the version and rank changes are zero, while other control messages are interchanged normally. While observing the power consumption on nodes with a clean network topology (Fig. 4), they tend to stabilize to 2% to 4%. This is considered ideal as the node is not being abused and hence can benefit from the long battery life.

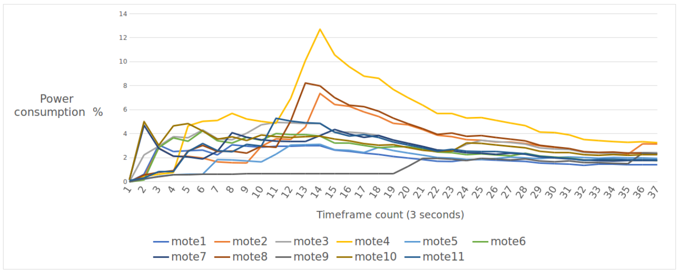


Fig. 4: Normal Power Consumption

1) *HELLO Flood Attack*

HELLO messages are used by nodes to announce themselves among their neighbors. They are broadcasted using DIO messages. An attacker might send DIO HELLO packets with increased power signal and a better routing metric, and hence introduce himself to many nodes as much as possible. Fig. 5 depicts the number of RPL control messages when the attack is launched.

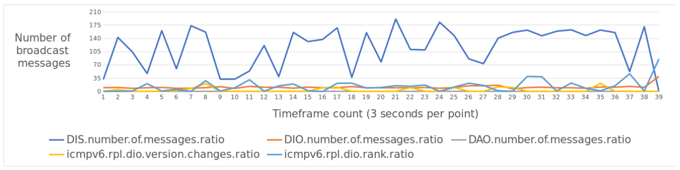


Fig. 5: RPL Control Messages Frequencies during a HELLO flood attack

It is clear that the number of DIS message is high in this attack. Thus, considering this type of control message later as an input parameter to our IDS will be beneficial.

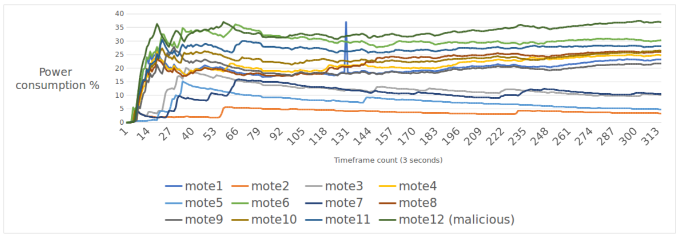


Fig. 6: Power Consumption per Mote during HELLO Flood

Regarding the power consumption during a HELLO flood attack (Fig. 6), the motes reached approximately 37% consumption which is a relatively high number compared to the clean topology. Hence, considering the power consumption of a mote as an input parameter is also considered as a good indicator.

2) Sinkhole Attack

The malicious attacker tries to create an artificial path to attract many reachable nodes and let them connect to it. This is achieved by announcing an optimal *rank* from the malicious node. As a result, traffic flowing through the neighbors will through it. This attack will not necessary disrupt the operation of the network, however, it might be combined with other attacks, and disrupt the topology.

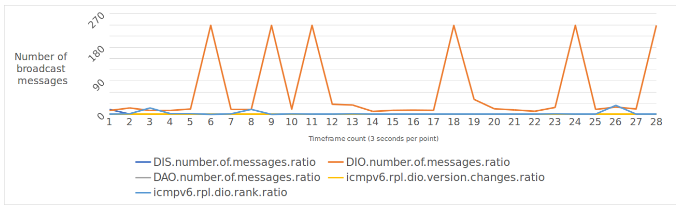


Fig. 7: RPL Control Messages Frequencies during a Sinkhole attack

In a sinkhole attack, the number of DIO messages seems to be the highest. This is due to the offers being sent by the motes on each rank change.

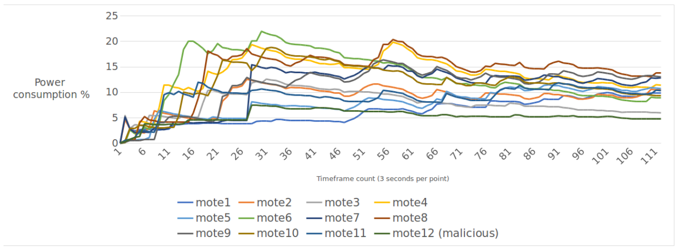


Fig. 8: Power Consumption per Mote during Sinkhole

The power consumption in a sinkhole attack is also considered high compared to the clean topology. This helps us later in differentiating between a malicious packet and a clean one.

3) Version Number Attack

The attacker node tries to modify the DODAG version by increasing its number. This will lead to an unnecessary rebuilding of the DODAG. When the root receives a DIO with an invalid version number, it resets its trickle timer for resending a new DIO.

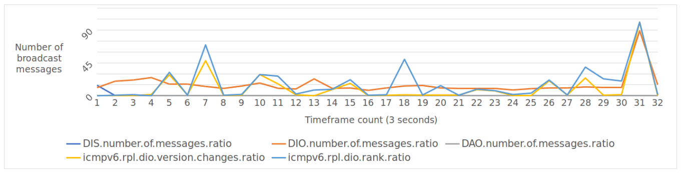


Fig. 9: RPL Control Messages Frequencies during a Version Number attack

Fig. 9 illustrates the version number attack RPL control messages frequencies. In this attack, we notice that the number of version changes is no longer zero. When combined with the power consumption of the node, we can detect that this is an anomalous behavior.

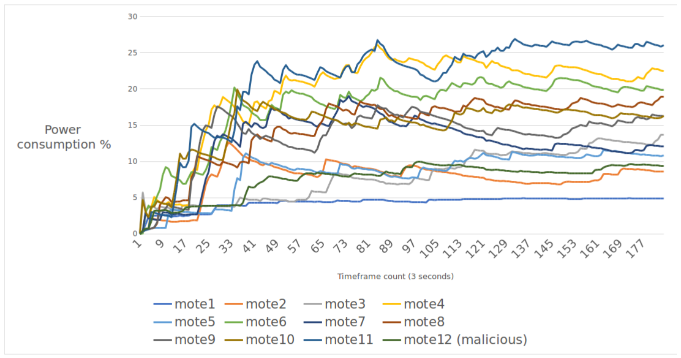


Fig. 10: Power Consumption per Mote during Version Number

The power consumption of the version number attack is also high compared to the clean topology. Hence, we can easily detect later the occurrence of an malicious activity.

There exist many other attacks on RPL, but in this paper we will only tackle the aforementioned attacks. A research in progress on how to integrate other attacks like Selective-forwarding[14], Sybil attack[15] and others. In the following subsection, we give an overview about the SOM and how it can be used to represent an input space of *n* dimensions to 2D space.

C. Self Organizing Map (SOM) Neural Networks

Self-Organizing Map, introduced by Prof. Kohonen in the 1980s is a type of neural network that uses unsupervised learning to convert from *nD* (high) dimensional space to low dimensional space that can be easily visualized on a *map*. SOM applies competitive learning through a neighborhood function to keep the topological properties of the input space. In a SOM topology, two fully connected layers exist: the input and output layers (see Fig. 2). A neighborhood relation is defined on the output neurons. As most learning algorithms, the weights are initialized randomly. An input vector is chosen at random and presented to the lattice. Every node calculates its distance to the input vector using a distance function. The node that is the closest to the input is considered as the Best Matching Unit (BMU). Then, a radius around this BMU is calculated, and all neighboring nodes' weights are adjusted to make them more like the input. The adjusted weights are calculated according to

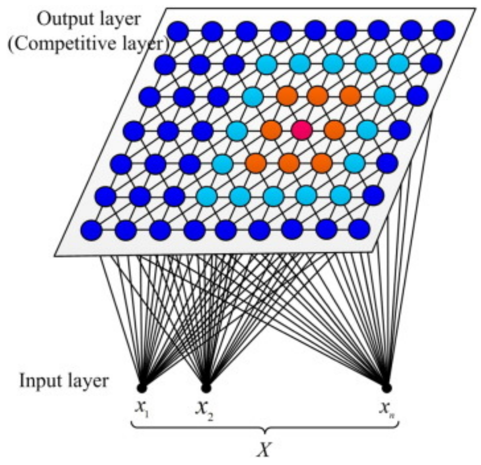


Fig. 11: SOM Topology (Architectural Graph)

Equation 1, where *t* is the time-step, *L* represents the learning rate which decreases with time.

$$W(t+1) = W(t) + \theta(t)L(t)(V(t) - W(t)) \quad (1)$$

The theta letter represents the influence rate a node's distance from the BMU has on its learning.

III. Proposed System

This section introduces the proposed system. We start with a high level architecture showing all major components involved from the learning phase till the *map* generation.

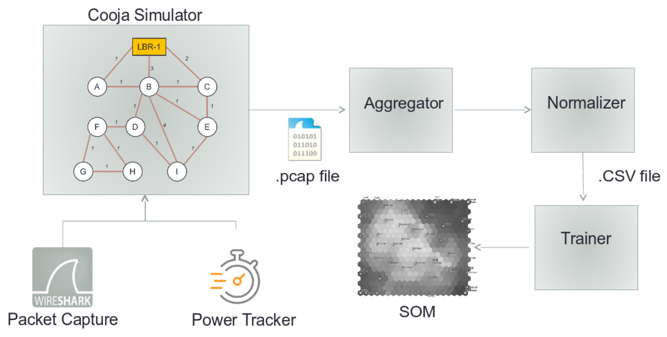


Fig. 12: High-level System Architecture

First, we start with simulating every attack mentioned earlier using the Cooja Simulator [16]. These attacks are implemented in [17] freely available on Github. For each simulation, we use Wireshark [18] to capture the traffic, and the Contiki's Power Tracker module to get the power consumption of every node during the attacks. The packet capture (pcap) file is fed into an aggregator which preprocess the data to eliminate noise and unnecessary packet fields. Then, the output of the aggregator is passed to a normalizer which normalizes the data between -1 and 1 for continuous variables and uses the one-of-N method for discrete variables. The normalized data (CSV file) is then fed into the trainer to produce a SOM map.

A. Network Design

We describe here the choice of all SOM parameters used in our system.

1) Measure of Distance

In SOM we determine the distance between the input vectors and the weights assigned to the neurons. Different distance functions exist such as Euclidean, Correlation, Cosine, Block Distance. In our system we selected the Euclidean distance (Equation 2).

$$dist(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (2)$$

2) Neighborhood Functions

The neurons in the lattice interact among themselves using a neighborhood function. Various functions exist such as the Mexican hat, cone, cylinder, Gaussian, etc). We decided to use the Gaussian function as a neighborhood function (Equation 3) in our SOM.

$$h(d) = e^{-(d^2/2.\sigma^2)} \quad (3)$$

3) Initialization of SOM

The weights in our SOM are initialized randomly, although there are different ways to initialize a SOM (Principal components, random samples from the input data). The number of neurons in the lattice is determined according to the Vesanto[19] formula (Equation 4).

$$msize = 5 * \sqrt{k} \quad (4)$$

The learning rate adopted is 0.5, which was selected empirically, till it reaches 0.01. The selected radius is 4 till it reaches 1. A decay function is also used which defines the decrease of the neighborhood size around the BMU (Equation 5).

$$\sigma(t) = \sigma_0 \exp(-\frac{t}{\lambda}) \quad (5)$$

4) Obtaining Data

Building a good model in machine learning is unquestionably dependent on the quantity, quality, preparation techniques, and relevance of the data obtained. By forming large datasets, the machine learning algorithm is exposed to a wider set of information that will help it learn, adapt, and gain knowledge. To form large datasets we relied on synthetic data resulting from multiple simulations with different scenarios using the COOJA network simulator for Contiki. The COOJA network simulator allows us to create real-life simulation scenarios, that captures and logs all packets that are exchanged between motes on the RPL protocol and one of the exclusive features of this simulator, is the ability to track the power consumption of each mote in near real-time. At the end of the simulation, we are left with a packet capture data file (.pcap), that can be fed into the Wireshark software for analysis and processing. To make it possible for our system to extract the data fields that are in interest for us, we need to export the packet capture data as XML format through the Wireshark software. We ran the simulation and collected 37061 packets for HELLO flood, 31248 packets for Sinkhole, 53410 packets for Version Number attack,and 25132 packets for clean data.

5) *Making Sense of the Data*

There are the six fields that are relevant for our proposed solution:

1) Icmpv6.code: The message type (DIS, DIO, DAO)
2) Ipv6.dst_host: The destination host IP address
3) Ipv6.src_host: The source host IP address
4) Icmpv6.rpl.dio.version: The current version for the DAG
5) Icmpv6.rpl.dio.rank: The current rank of the source
6) Timestamp: The Unix timestamp

Table I shows an example of a packet data. Looking at the table's record, we are not able to conclude whether this packet is from a malicious source or not, but having a set of records would help us decide if for a certain time window an intrusion did take place. After parsing the XML data exported from the Wireshark software, it is going to be passed as input to our aggregator module. The aggregator module will morph the previously mentioned packet fields into six different features by aggregating the data in time window of three seconds while grouping on the destination host, i.e: All packets for a destination host that are in the range of the three seconds timeframe will be aggregated and transformed into a vector of six features. To enhance the relation between the data fields, the features that we propose are the following:

1) DIS.number.of.messages.ratio: The number of DIS messages with respect to the total number of messages in the timeframe for the destination host.
2) DIO.number.of.messages.ratio: The number of DIO messages with respect to the total number of messages in the timeframe for the destination host.
3) DAO.number.of.messages.ratio: The number of DAO messages with respect to the total number of messages in the timeframe for the destination host.
4) Icmpv6.rpl.dio.version.changes.ratio: The ratio of version changes with respect to the number of messages in the timeframe for the destination host.
5) Icmpv6.rpl.dio.rank.ratio: The ratio of rank changes with respect to the number of messages in the timeframe for the destination host.
6) Mote.avg.power: The average power consumption for the destination mote in the timeframe, has the value 0 if the destination is a broadcast address.

Table II shows an example of aggregated data.

6) *Making the data uniform*

In the previous example, the aggregated data does not seem to be uniform, the values might range between 0 and 400 in the case of a three seconds time frame; widening that window may result in even a bigger gap between the minimum and maximum values. One way to bridge the range gap is to normalize the data between -1 and 1. Our normalization of choice is the min-max normalization which is known as Feature Scaling (Equation 6).

$$X' = \frac{(x - dl)(vh - vl)}{(dh - dl)} \tag{6}$$

where:

x	= The current value
d	= The high and low values for x
v	= The high and low desired values for normalization

IV. Results and Simulation

As mentioned earlier, Cooja is used as the simulation environment. Wireshark extension is integrated in Cooja to capture IEEE 802.15.4 packets.

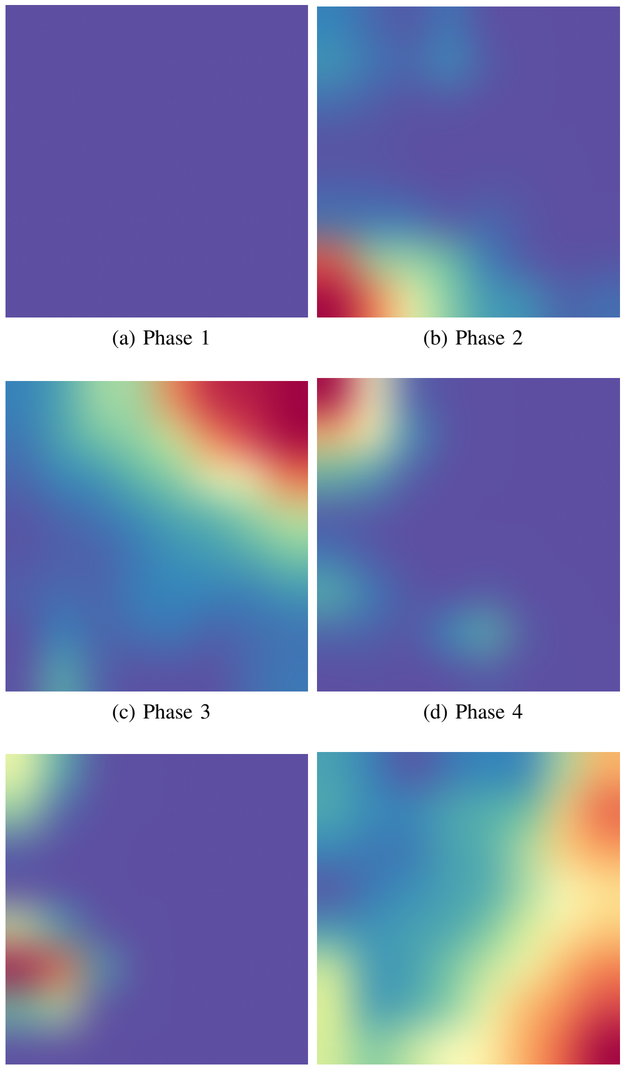


Fig. 13: SOM Formation Steps

The Powertrace [20] module which is built-in into Cooja is used to track the power of the motes. Regarding the SOM implementation, SOMOCLU [21], which is a massively

TABLE I: Packet Data Example

icmpv6.code	ipv6.dst_host	ipv6.src_host	icmpv6.rpl.dio.version	icmpv6.rpl.dio.rank	Timestamp
DIO	ff02000000000000 000000000000001a	fe80000000000000 c30c000000000000	f0	80	1515578910.8592

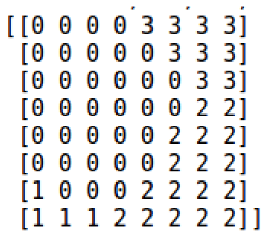
TABLE II: Aggregated Data Example

DIS.number.of. messages.ratio	DIO.number.of. messages.ratio	DAO.number.of. messages.ratio	icmpv6.rpl.dio. version.changes.ratio	icmpv6.rpl.dio. rank.ratio	mote.avg.power
183.0274	19.471	0	0.649033	0.649033	22.56
59.05345	17.15839	0	0.500453	32.02899	36.75
0	0	285.4261	0	0	23.62
0	34.19606	79.79081	0.759912	0	22.56

TABLE III: Normalized Data Example

DIS.number.of. messages.ratio	DIO.number.of. messages.ratio	DAO.number.of. messages.ratio	icmpv6.rpl.dio. version.changes.ratio	icmpv6.rpl.dio. rank.ratio	mote.avg.power
0.004786	-0.94739	-1	-0.99647	-0.94775	0.822006
0.172729	0.94103	-1	-0.88365	-0.99347	0.848834
-1	-0.9858	-0.90598	-0.99201	-1	-0.62798

parallel SOM Python library is used. Matplotlib [22] Python library is used to plot the SOM each phase during training and its final state (Fig. 13). After training, a U-Matrix is generated (Fig. 14). The U-Matrix (unified distance matrix) is a representation of a self-organizing map (SOM) where the Euclidean distance between the codebook vectors of neighboring neurons is depicted in a grayscale image. This figure is used to visualize the data in a high-dimensional space using a 2D image [23].



```
[[0 0 0 0 3 3 3 3]
 [0 0 0 0 0 3 3 3]
 [0 0 0 0 0 0 3 3]
 [0 0 0 0 0 0 2 2]
 [0 0 0 0 0 2 2 2]
 [0 0 0 0 0 2 2 2]
 [1 0 0 0 2 2 2 2]
 [1 1 1 2 2 2 2 2]]
```

Fig. 14: Resulting U-Matrix of the Training

The visualization of the relative component distributions of the input data can be achieved using SOM Component Planes (Fig. 15). We can easily notice that we have four classes as output: Clean, HELLO Flood, Sinkhole, and Version Number. The SOM labeling can be summarized as follows:

1) Top-left cluster represents the clean packets.
2) Top-right cluster represents the Sinkhole attack.
3) Bottom-right cluster represents the HELLO flood attack.
4) Bottom-left cluster represents the Version number change attack.

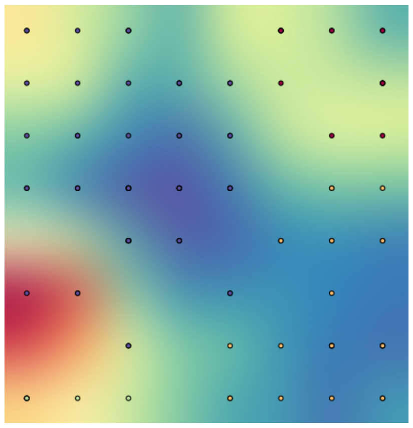


Fig. 15: SOM Component Plane

V. Conclusion

The RPL protocol was developed to help reduce power consumption on IoT devices; Under the perfect circumstances the life-span of the IoT devices can be prolonged up-to a decade, but as intrest grow, the need for security grows even further. With the pioneering of Machine Learning models, engineers tried to combine some of the most powerful neural networks with absurd and non-linear problems to obtain instant and error-prone results, allowing the system to invoke actions accordingly. The detection and classification of malicious attacks on an RPL network is important to help preserve the network's integrity. To recognize the abnormal behaviour at an early stage, we've trained the Self-Organizing Map using an unsupervised learning model. By aggregating the networks' packet capture along with the average power consumption or each mote on a specified timeframe, the SOM was able to cluster the normalized input training samples into four distinctive classes: HELLO Flood Attack, Sinkhole Attack, Version Attack, and clean data. As part of a future study, the IDS will be able to detect additional attacks, and the resulting SOM UMatrix would be used with fuzzy classification such as the Fuzzy C-means clustering.

REFERENCES

[1] Elie Nasr, Elie Kfoury, and David Khoury. An IoT approach to vehicle accident detection, reporting, and navigation. In *Multidisciplinary Conference on Engineering Technology (IMCET), IEEE International*, pages 231–236. IEEE, 2016.

[2] Security and costs holding back those looking to implement IoT projects, https://www.helpnetsecurity.com/2017/12/05/implement-iot-projects. 2017

[3] Altman Vilandrie & Company Are your company's IoT devices secure? 2017.

[4] David Khoury and Elie Kfoury. Generic hybrid methods for secure connections based on the integration of GBA and TLS/CA. In *2017 Sensors Networks Smart and Emerging Technologies (SENSET)*. IEEE, sep 2017.

[5] Shahid Raza, Thiemo Voigt, and Vilhelm Jutvik. Lightweight ikev2: a key management solution for both the compressed ipsec and the ieee 802.15. 4 security. In *Proceedings of the IETF workshop on smart object security*, volume 23, 2012.

[6] Shahid Raza, Linus Wallgren, and Thiemo Voigt. Svelte: Real-time intrusion detection in the internet of things. *Ad hoc networks*, 11(8):2661–2674, 2013.

[7] Sophia Kaplantzis, Alistair Shilton, Nallasamy Mani, and Y Ahmet Sekercioglu. Detecting selective forwarding attacks in wireless sensor networks using support vector machines. In *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, pages 335–340. IEEE, 2007.

[8] M Ahmadi Livani and Mahdi Abadi. A pca-based distributed approach for intrusion detection in wireless sensor networks. In *Computer Networks and Distributed Systems (CNDS), 2011 International Symposium*, 2011.

[9] Roger Achkar, Mustafa El-Halabi, Elie Bassil, Rayan Fakhro, and Marny Khalil. Voice identity finder using the back propagation algorithm of an artificial neural network. *Procedia Computer Science*, 95:245–252, 2016.

[10] Youssef Harkouss, Souhad Mcheik, and Roger Achkar. Accurate wavelet neural network for efficient controlling of an active magnetic bearing system. 2010.

[11] George Abou Kassm and Roger Achkar. Lpr cnn cascade and adaptive deskewing. *Procedia Computer Science*, 114:296–303, 2017.

[12] Chafic Saide, Régis Lengelle, Paul Honeine, Cédric Richard, and Roger Achkar. Nonlinear adaptive filtering using kernel-based algorithms with dictionary adaptation. *International Journal of Adaptive Control and Signal Processing*, 29(11):1391–1410, 2015.

[13] P Thubert, A Brandt, J Hui, R Kelsey, P Levis, K Pister, R Struik, JP Vasseur, and R Alexander. Rpl: Ipv6 routing protocol for low power and lossy networks. *RFC 6550*, 2012.

[14] Leela Krishna Bysani and Ashok Kumar Turuk. A survey on selective forwarding attack in wireless sensor networks. In *Devices and Communications (ICDeCom), 2011 International Conference on*, pages 1–5. IEEE, 2011.

[15] Faiza Medjek, Djamel Tandjaoui, Mohammed Riyadh Abdmeziem, and Nabil Djedjig. Analytical evaluation of the impacts of sybil attacks against rpl under mobility. In *Programming and Systems (ISPS), 2015 12th International Symposium on*, pages 1–9. IEEE, 2015.

[16] Fredrik Osterlind. A sensor network simulator for the contiki os. *Swedish Institute of Computer Science (SICS), Tech. Rep. T2006-05*, 2006.

[17] RPL Attacks Framework, https://github.com/dhondta/rpl-attacks/. 2017

[18] Wolf-Bastian Pöttner and Lars Wolf. Ieee 802.15. 4 packet analysis with wireshark and off-the-shelf hardware. In *Proceedings of the Seventh International Conference on Networked Sensing Systems (INSS2010). Kassel, Germany*, 2010.

[19] Jochen Wendel and Barbara P Buttenfield. Formalizing guidelines for building meaningful self-organizing maps. *GIScience 2010 Short Paper Proceedings, Zurich, Switzerland, September*, 2010.

[20] Adam Dunkels, Joakim Eriksson, Niclas Finne, and Nicolas Tsiftes. Powertrace: Network-level power profiling for low-power wireless networks, 2011.

[21] Peter Wittek, Shi Chao Gao, Ik Soo Lim, and Li Zhao. Somoclu: An efficient parallel library for self-organizing maps. *arXiv preprint arXiv:1305.1422*, 2013.

[22] John D Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[23] Alfred Ultsch. Kohonen's self organizing feature maps for exploratory data analysis. In *Proceedings INNC'90, International Neural Network Conference, 1990*, pages 305–308. Kluwer, 1990.