A Central Intrusion Detection System for RPL-Based Industrial Internet of Things

Conference Paper · May 2019

DOI: 10.1109/WFCS.2019.8758024

6 authors, including:

Emre Aydoğan
Akdeniz University
4 PUBLICATIONS **21** CITATIONS
SEE PROFILE

Sevil Sen
Hacettepe University
40 PUBLICATIONS **518** CITATIONS
SEE PROFILE

Ismail Butun
Chalmers University of Technology
45 PUBLICATIONS **809** CITATIONS
SEE PROFILE

A Central Intrusion Detection System for RPL-Based Industrial Internet of Things

Emre Aydogan*, Selim Yilmaz*,
and Sevil Sen
WISE Lab., Department of Computer Engineering
Hacettepe University, Ankara, Turkey
Contact E-mail: ssen@cs.hacettepe.edu.tr

Ismail Butun, Stefan Forsström,
and Mikael Gidlund
Department of Information Systems and Technology
Mid Sweden University, Sundsvall, Sweden
Contact E-mail: ismail.butun@miun.se

***Abstract*—Although Internet-of-Things (IoT) is revolutionizing the IT sector, it is not mature yet as several technologies are still being offered to be candidates for supporting the backbone of this system. IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) is one of those promising candidate technologies to be adopted by IoT and Industrial IoT (IIoT). Attacks against RPL have shown to be possible, as the attackers utilize the unauthorized parent selection system of the RLP protocol. In this work, we are proposing a methodology and architecture to detect intrusions against IIoT. Especially, we are targeting to detect attacks against RPL by using genetic programming. Our results indicate that the developed framework can successfully (with high accuracy, along with high true positive and low false positive rates) detect routing attacks in RPL-based Industrial IoT networks.**

***Index Terms*—RPL attacks, security, intrusion detection, IoT, genetic programming.**

I. Introduction

Today we see applications that can efficiently utilize information from sensors that are attached to things in order to provide more automatized, rational, and intelligent behavior. This concept is more commonly being referred to as the Internet-of-Things (IoT) [1] in recent years, where IoT is the common notion for the development of the things (machinery, vehicles, goods, appliances, clothes, etc.) by having equipped with small embedded sensors and actuators so that they can also communicate among each other over the Internet. This leads those devices to perceive their surroundings, to have intelligent behavior, to communicate with each other and further apart parties, and finally to create new forms of smart services that are useful for human beings. From this we also see various technological trends such as integration of IoT with cloud computing for large scale data storage, big data analysis on massive amount of gathered data from IoT resources, incorporation of cyber-physical systems into machine to machine (M2M) systems. In relation to all stated above, there is the Industrie 4.0 (Industry 4.0) initiative, which includes smart cities, smart industry, factories of the future, and smart manufacturing; forming what we see today as the research area of Industrial IoT (IIoT) [2].

Industrial sector demands are quite different from non IIoT vendors, especially when it comes to being time critical and being reliable [3]. For example, an industrial process might have to react promptly to small changes in the sensor values to maintain a high quality of the product or to avoid a catastrophic failure. Owing to this fact, industrial communication systems often consider a notion called "five nines availability" [4], meaning that an uptime of the system to be at least 99.999%. In addition to this, industrial applications and IIoT have much higher security demands, to avoid downtime and to protect sensitive information related to the industrial process against industrial espionage [5]. Devised security measures should protect the industrial networks from various attacks such as denial of service (DoS), as well as should provide data protection, privacy of the sensitive industrial data, and timely updates of the network components to avoid exploitation. In this work, we are investigating to find remedies in securing IIoT networks from the adversaries and ill mannered attackers against industrial systems.

In the literature, nature-inspired algorithms are extensively used for solving various real world problems such as optimization, control systems, robotics, and etc. Among them, genetic programming is one of the most popular evolutionary computation-based algorithms. Here, in this work, our aim is to utilize genetic programming for cyber-security of IIoT networks in detecting attacks towards IIoT and to propose an Intrusion Detection System (IDS). The research work presented in this paper seeks to answer the following three research questions, in the area of intrusion detection for the IIoT networks:

1) How can genetic programming be used to detect attacks against the IIoT networks?
2) How can a practical implementation of such a system be realized? How well does such an implementation perform in terms of detection success rate and detection time?
3) What is the suitable intrusion detection architecture for IIoT?

From these research questions, our contribution in this paper includes an intrusion detection framework that relies on genetic programming for detecting intrusions in IIoT systems, as well as a proof of concept implemented and a quantitative evaluation thereof. Moreover, the suitability of a central

* Authors have equal contribution.

intrusion detection system at the root node is explored, since the network contains various devices with different capabilities and resource constraints. It is shown that the root node could detect attacks against RPL effectively in a timely manner.

The remainder of this article is organized as follows: Section II presents relevant background and related work. Section III presents the general approach and the details of the methodology that is proposed for detecting intrusions by using genetic programming. Section IV presents the proof-of-concept implementation. Section V presents the performance evaluation results that are obtained through simulations. Finally, Section VI concludes.

II. Background

IoT is revolutionizing the IT sector and its all sub-dependent sectors by providing remote sensing and control to be available to the users that are located at far distances from the target position. These things (devices, things, and nodes will be used interchangeably in this text to represent the smallest unit in IoT) might be as simple as sensing devices such as proximity sensors, smart locks, IP cameras, and weather stations; to more complex systems such as security systems, monitoring systems, and factory automation systems (for IIoT) [6]. IIoT is expected to constitute one of the main pillars during the Industry 4.0 revolution by helping seamlessly implementation of remote sensing (through sensors, etc.) and acting (through actuators, etc.) for factory automation systems [7].

Although there are many application areas and also many users of IoT (and IIoT) throughout the world, it is not mature yet as several technologies are still being offered to be a candidate architecture to support the backbone of this system. For instance, TelosB/Sky is an IEEE 802.15.4 compliant wireless sensor mote based on the open source TinyOS operating system and widely adopted by the IoT/IIoT researchers [8]. It has been shown in the literature that TelosB/Sky mote is a very good candidate for IIoT applications, especially through transitioning from IoT towards IIoT [9]. Therefore, in order to provide a proof of concept on IIoT networks, we have run the RPL routing protocol on top of the TelosB/Sky motes through the simulations and testing of our proposed IDS scheme.

Routing Protocol for Low-Power and Lossy Networks (RPL) is also one of the promising candidate technologies to be adopted by IoT. Attacks against RPL have shown to be possible, as the attackers utilize the unauthorized parent selection mechanism. Moreover, local-repair mechanism of DODAG can be exploited to execute routing attacks. During global-repair mechanism of DODAG, two versions of DODAG tree exist which can be exploited by the attackers. Version numbering system (version numbers) of the RPL DODAG tree is not controlled by a centralized or a distributed authority, therefore it is also susceptible to attacks.

In the literature, there are a few papers targeted at detecting or mitigating attacks related to RPL. Here, we especially focus on studies that aim to detect hello flood and version attacks particularly as in the current study. Napiah *et al.* [10] proposed

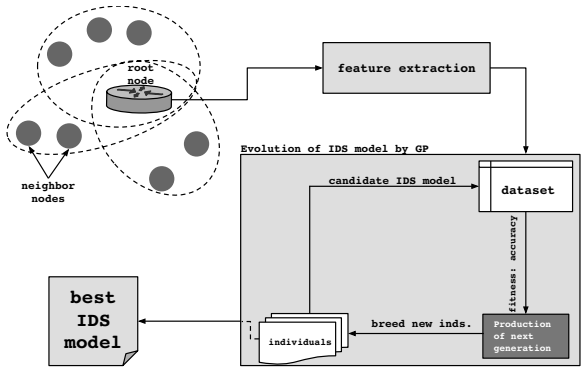

Fig. 1. An overview of GP-based IDS approach.

an IDS known as Compression Header Analyzer-Intrusion Detection System (CHA-IDS) that analyzes 6LoWPAN compression header data to mitigate the individual and combination routing attacks against RPL protocol. The proposed CHA-IDS scheme has shown to be successful against routing attacks; hello flood, sinkhole, and wormhole.

Mayzaud *et al.* [11] investigated the detection of version number attacks against RPL protocol of IoT. In their work, they identified several metrics to be deeply affected by this kind of attack. These metrics are provided as: control packet overhead, packet delivery ratio, and end-to-end delay. Ahmed and Ko [12] proposed a distributed and cooperative verification mechanism to securely defend against the DODAG version number attack with low control overhead and high reliability.

Aris *et al.* [13] proposed two mitigation techniques for the version number attack with different performance outcomes and resource requirements. The first technique eliminates the version number updates coming from the direction of the leaf nodes. By this way, it makes sure that the strongest positions for the version number attacks are mitigated. However, for the rest of the attacking positions, it cannot mitigate the effects of the attack. Therefore, they proposed a second mitigation technique in order to mitigate the effect of the attack regardless of the attacking positions. Here a node is allowed to change its version number only when the majority of its neighbors with better ranks claim a version number update.

III. Proposed Detection Methodology

Here in this research, we explore the usage of genetic programming in detection of attacks against the RPL protocol in IIoT networks. As mentioned earlier in this text, RPL's DODAG mechanism is vulnerable to many attacks. Our aim here is to propose a suitable intrusion detection system for IIoT. In order to address this problem, we explore the use of genetic programming (GP). Evolutionary-computation based techniques are known to be quite effective in intrusion detection [14] and able to discover characteristics of complex networks such as ad hoc and sensor networks [15]. Therefore, their potentials in intrusion detection in IIoT networks are searched for in this study.

Genetic programming [16] is inspired from the mechanism of natural evolution, which is based on that the individuals best suited for the environment have a higher chance of surviving through generations. Similarly, in the evolutionary computation, proposed candidate solutions for a problem called individuals are evolved by using genetic operators such as crossover and mutation in order o produce better solutions over time.

In genetic programming, individuals are represented by trees. In this study, the in-order traversal of a tree (individual) outputs an if statement for detecting RPL attacks. An example to a simple GP tree is given in Fig. 2. The in-order traversal statement corresponding to this tree is `if(COUNT_DIO < COUNT_DIS) and (COUNT_DATA == (0.99 * AVG_TIME_BTW_DATA))`. If this statement is satisfied, an attack alarm is raised.

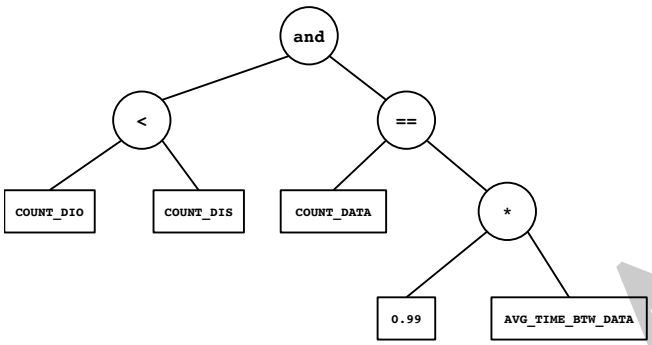

Fig. 2. An example of a GP tree.

The general steps of genetic programming algorithm are given in Algorithm 1. Please note that the individuals in the first population are randomly generated. In crossover operation, subtrees of selected individuals from the current population are exchanged. In mutation operation, a subtree of a selected individual is exchanged with a random subtree in order to maintain diversity in the population. For these operations, the individuals are selected based on their fitness values which show how well they solve the problem. These steps are applied over generations and the output of the algorithm is either the ideal solution or the best individual produced after a number of generations. The general schema of the proposed approach is depicted in Fig. 1.

Initialize population;
repeat
  Evaluate the fitness of each individual;
  Rank the population according to fitness values;
  Apply generic operators (crossover, mutation etc.) and reproduce new population;
until *termination criterion is satisfied*;
return best-of-run individual;

Algorithm 1: The general steps of GP

In order to decrease the computation and communication overhead on nodes responsible for intrusion detection, a central intrusion detection architecture is proposed. In this architecture, only the root node is responsible for making decisions about attacks. The root note monitors packets in his neighbourhood and extracts some features from these packets periodically. The features are listed in Table II. The features employed in this study are more comprehensive than other studies in the literature. Please note that these features together with some mathematical and logical operators given in Table I constitute GP trees. In this study, the first population is initially initialized. The evolution, which generates the intrusion detection algorithm automatically, takes place on the root node offline and continues until the 1000th generation is reached. Then, the best individual at the last generation is evaluated on the testing network. Genetic programming is run for each attack type separately, so two detection algorithms are obtained at the end of the evolution process. The GP parameters used in this study are listed in Table I. The parameters not listed here are the default parameters of the ECJ tool [17].

TABLE I
GP PARAMETERS

Parameters	Value
Functions	+, -, *, /, sin, cos, log, ln, sqrt, abs, exp, ceil, floor, max, min, pow, mod, $<$, $\leq$, $>$, $\geq$, $==$, $!=$, and, or
Terminals	rnd(0,1) and features
Population Size	50 (number of elite individuals = 5)
Crossover probability	0.9
Mutation probability	0.1
Selection strategy	Tournament selection (Tournament size: 7)
Generations	1000

IV. IMPLEMENTED SYSTEM

We have used 26 nodes IIoT network that are comprised of Telos-B modes each operating Contiki O.S. [18] on them. The network is constructed and simulated on Cooja simulation environment [19]. Each simulation is run for almost 20 minutes. One of the nodes acts as a *root* node working as a data-sink, so that all the network packets are destined to it. Training and testing networks are set-up in a randomly distributed network topology. Around $\approx$10% of the nodes are selected as attackers (non-root ones) which implement the following two attack types:

1) *Hello flood attack:* The RPL routing protocol is susceptible to hello flood attacks [20]. Each node in RPL routing protocol broadcasts "HELLO" message to inform of their presence to their one hop neighbors. A malicious node floods "HELLO" packets with high enough transmission power to convince every node in the network that it is their neighbor. In the attack scenario, all attackers perform this attack at the same time.
2) *Version number attack:* In version number attack, attackers manipulate their own DODAG version number to manipulate RPL's "global repair" mechanism, so that

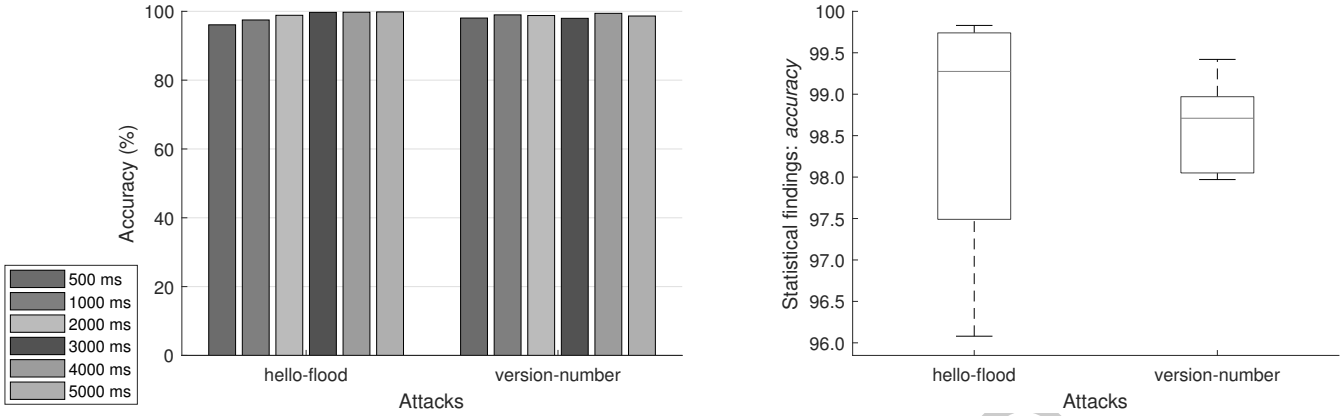

Fig. 3. Accuracy of the proposed IDS under hello flood and version number attacks.

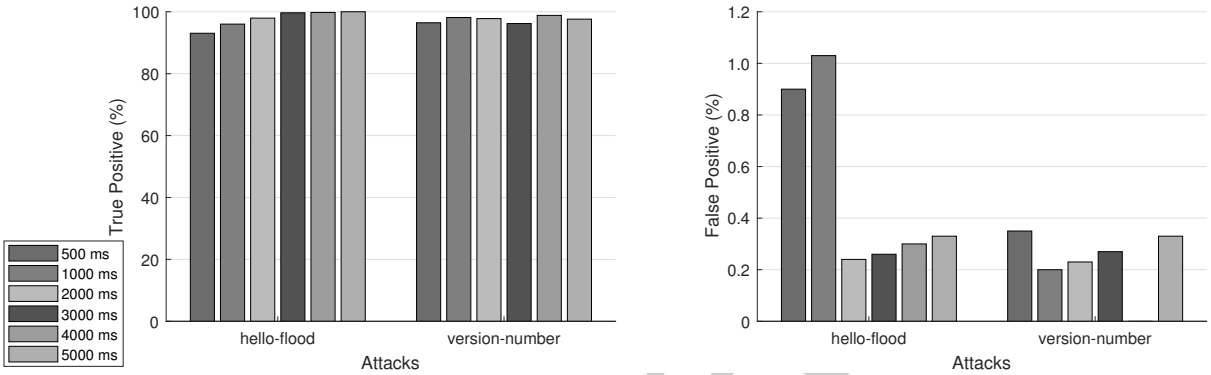

Fig. 4. True positive vs. false positive rates of the proposed IDS under hello flood and version number attacks.

they gain privilege in the network. In our implementation, all of the malicious nodes execute this attack at the same time.

The root node computes the features at periodic time intervals. In order to find the ideal detection interval, the features are collected at different time intervals: 500 ms, 1000 ms, 2000 ms, 3000 ms, 4000 ms, and 5000 ms. While the time interval increases, the computation overhead decreases at the root node. However all packets received in that period are needed to be buffered. So it is a trade-off to make between the memory consumption and the computation overhead. Therefore, the most suitable period could be selected based on the root device's physical characteristics.

V. Measurements and Evaluation

According to the extensive simulation results, the proposed IDS solution based on genetic programming performs with promising results. The accuracy of the proposed IDS under two different types of attacks are presented in Fig. 3. Whereas, Fig. 4 presents true positive vs. false positive rates of the proposed IDS under hello flood and version number attacks for a further comparisons.

As it is seen in the results, both attacks could be detected effectively enough by employing lower intervals. Hence, an attack could be detected before harming the network further. In addition, it can easily deduced from the results, the proposed IDS has shown better detection performance as the time interval increases for the hello flood attack. For this attack type, 96.08% and 99.83% are the worst and best accuracy values obtained with collecting data in 500 ms and 5000 ms intervals, respectively. As for the version number attack, it is seen that the proposed IDS has shown closer performance than that in hello flood attack in different time intervals. The best and the worst accuracy has been obtained with the intervals of 4000 ms (99.42%) and 3000 ms (97.97%), respectively.

These results represent the effectiveness of a centralized detection system as the evolved intrusion detection algorithm is only placed at the root node. However, the proposed framework can easily be adopted to any dedicated node as well. In this study, a distributed intrusion detection architecture, where every node is responsible for local detection based on the packets in their neighbourhoods, is also analyzed. In the distributed architecture, 51% and 71% of all nodes detect hello flood and version number attacks respectively with an accuracy higher than 90%. Especially for the hello flood attack, the root node performs better than a distributed architecture, since it collects more information than individual nodes. Moreover

this architecture is more suitable for IoT, since some nodes might not have enough capabilities and resources to perform intrusion detection for IoT. The only downside of the proposed architecture is having the single point of failure. To avoid that, one of the root nodes' neighbours could take place of the root node in case of a failure.

VI. Conclusion

This article investigates a suitable intrusion detection system for RPL-based IIoT and answers three research questions. In the first question, the usage of genetic programming for detecting attacks against RPL is explored. The second question is on the realization of the proposed systems. The simulations show that the proposed IDS can detect intrusions with very satisfying results (high true positive and low false positive rates). For the final question, a central intrusion detection system, which is more suitable for this environment, has compared with a distributed architecture and shown to be very effective in detecting attacks in a timely manner.

Appendix

The feature set used for GP tree construction is listed in Table II.

TABLE II
THE FEATURE SET

No	Feature	Description
1	COUNT_DATA	Number of data packets
2	COUNT_DIO	Number of DIO packets
3	COUNT_DIS	Number of DIS packets
4	COUNT_DAO	Number of DAO packets
5	COUNT_DAOACK	Number of DAOACK packets
6	MAX_VERSION	Max. of versions
7	MIN_VERSION	Min. of versions
8	AVG_VERSION	Avg. of versions
9	MAX_RANK	Max. of ranks
10	MIN_RANK	Min. of ranks
11	AVG_RANK	Avg. of ranks
12	MAX_INTERVAL_MIN	Max. of DIO interval min.
13	MIN_INTERVAL_MIN	Min. of DIO interval min.
14	AVG_INTERVAL_MIN	Avg. of DIO interval min.
15	MAX_INTERVAL_DOUB	Max. of DIO interval doublings
16	MIN_INTERVAL_DOUB	Min. of DIO interval doublings
17	AVG_INTERVAL_DOUB	Avg. of DIO interval doublings
18	MAX_RPL_INS	Max. of rpl instances
19	MIN_RPL_INS	Min. of rpl instances
20	AVG_RPL_INS	Avg. of rpl instances
21	MAX_DATA_LEN	Max. length of data packets
22	MIN_DATA_LEN	Min. length of data packets
23	AVG_DATA_LEN	Avg. length of data packets
24	MAX_DIO_LEN	Max. length of DIO packets
25	MIN_DIO_LEN	Min. length of DIO packets
26	AVG_DIO_LEN	Avg. length of DIO packets
27	MAX_DIS_LEN	Max. length of DIS packets
28	MIN_DIS_LEN	Min. length of DIS packets
29	AVG_DIS_LEN	Avg. length of DIS packets
30	MAX_DAO_LEN	Max. length of DAO packets
31	MIN_DAO_LEN	Min. length of DAO packets
32	AVG_DAO_LEN	Avg. length of DAO packets
33	MAX_DAOACK_LEN	Max. length of DAOACK packets
34	MIN_DAOACK_LEN	Min. length of DAOACK packets
35	AVG_DAOACK_LEN	Avg. length of DAOACK packets
36	MAX_TIME_BTW_DATA	Max. time between data packets
37	MIN_TIME_BTW_DATA	Min. time between data packets
38	AVG_TIME_BTW_DATA	Avg. time between data packets
39	MAX_TIME_BTW_DIO	Max. time between DIO packets
40	MIN_TIME_BTW_DIO	Min. time between DIO packets
41	AVG_TIME_BTW_DIO	Avg. time between DIO packets
42	MAX_TIME_BTW_DIS	Max. time between DIS packets
43	MIN_TIME_BTW_DIS	Min. time between DIS packets
44	AVG_TIME_BTW_DIS	Avg. time between DIS packets
45	MAX_TIME_BTW_DAO	Max. time between DAO packets
46	MIN_TIME_BTW_DAO	Min. time between DAO packets
47	AVG_TIME_BTW_DAO	Avg. time between DAO packets
48	MAX_TIME_BTW_DAOACK	Max. time between DAOACK packets
49	MIN_TIME_BTW_DAOACK	Min. time between DAOACK packets
50	AVG_TIME_BTW_DAOACK	Avg. time between DAOACK packets

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[3] J. Åkerberg, M. Gidlund, and M. Björkman, "Future research challenges in wireless sensor and actuator networks targeting industrial automation," in *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*. IEEE, 2011, pp. 410–415.

[4] I. Silva, L. A. Guedes, P. Portugal, and F. Vasques, "Reliability and availability evaluation of wireless sensor networks for industrial applications," *Sensors*, vol. 12, no. 1, pp. 806–838, 2012.

[5] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial internet of things," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*. IEEE, 2015, pp. 1–6.

[6] M. Kocakulak and I. Butun, "An overview of wireless sensor networks towards internet of things," in *Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual*. IEEE, 2017, pp. 1–6.

[7] S. Forsström, I. Butun, M. Eldefrawy, U. Jennehag, and M. Gidlund, "Challenges of securing the industrial internet of things value chain," in *2018 Workshop on Metrology for Industry 4.0 and IoT*. IEEE, 2018, pp. 218–223.

[8] J. Williams. (February 2019) Tmote sky & tinyos. https://telosbsensors.wordpress.com/2013/12/31/tmote-sky-tinyos/.

[9] F. Dobslaw, T. Zhang, and M. Gidlund, "Qos-aware cross-layer configuration for industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 5, pp. 1679–1691, 2016.

[10] M. N. Napiah, M. Y. I. B. Idris, R. Ramli, and I. Ahmedy, "Compression header analyzer intrusion detection system (cha-ids) for 6lowpan communication protocol," *IEEE Access*, vol. 6, pp. 16 623–16 638, 2018.

[11] A. Mayzaud, R. Badonnel, and I. Chrisment, "Detecting version number attacks in rpl-based networks using a distributed monitoring architecture," in *Network and Service Management (CNSM), 2016 12th International Conference on*. IEEE, 2016, pp. 127–135.

[12] F. Ahmed and Y.-B. Ko, "A distributed and cooperative verification mechanism to defend against dodag version number attack in rpl," in *Proceedings of the 6th International Joint Conference on Pervasive and Embedded Computing and Communication Systems*. SCITEPRESS-Science and Technology Publications, Lda, 2016, pp. 55–62.

[13] A. Ar, S. B. rs Yaln, and S. F. Oktu, "New lightweight mitigation techniques for rpl version number attacks," *Ad Hoc Networks*, vol. 85, pp. 81 – 91, 2019.

[14] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Applied soft computing*, vol. 10, no. 1, pp. 1–35, 2010.

[15] S. Sen, "A survey of intrusion detection systems using evolutionary computation," in *Bio-inspired computation in telecommunications*. Elsevier, 2015, pp. 73–94.

[16] J. R. Koza and J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992, vol. 1.

[17] ECJ. (February 2019) A java-based evolutionary computation research system. https://cs.gmu.edu/~eclab/projects/ecj/.

[18] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *29th annual IEEE international conference on local computer networks*. IEEE, 2004, pp. 455–462.

[19] F. Österlind, "A sensor network simulator for the contiki os," *SICS Research Report*, 2006.

[20] P. Pongle and G. Chavan, "A survey: Attacks on rpl and 6lowpan in iot," in *Pervasive Computing (ICPC), 2015 International Conference on*. IEEE, 2015, pp. 1–6.