IJMSIT

International Journal of Multidisciplinary Studies and Innovative Technologies

Volume : 5
Number: 2
Year: 2021
Pages: 123-128

Enhancement Trust Management in IoT to Detect ON-OFF Attacks with Cooja

Ali Hamid Farea[1*] and Kerem Küçük [2#]

[1]*Department of Computer Engineering, Kocaeli University, Kocaeli, Turkey*
[2]*Department of Software Engineering, Kocaeli University, Kocaeli, Turkey*
[*] 195112025@kocaeli.edu.tr
[#] kkucuk@kocaeli.edu.tr

***Abstract* –** In IoT ecosystems, the interaction of devices with each other creates a perfect environment but there are heterogeneous nodes that will supply a variety of services. In the intelligent environment, devices with various processing capacities may operate together and communicate transparently with one other and with users. These IoT gadgets are frequently exposed to the public and interact over wireless channels, making them vulnerable to malicious attacks. ON-OFF attacks (OOAs) are regarded as one of the IoT's trust threats. In these attacks, the malicious nodes alternate between behaving well and behaving badly, jeopardizing the network if they stay trusted nodes. In this paper, we introduce a model to enhance trust management in IoT to detect (OOAs) with the help of Artificial Neural Networks (ANN) to analyze the statuses (ON-OFF) and radio messages for each node which in turn assesses the resource trust automatically in IoT. We implemented our experiment by using Contiki Operating System (OS) and analyzed the data with Microsoft machine learning studio (MMLS) to display the results.

Keywords – Trust management, IoT, ON-OFF attacks, Artificial Neural Networks, Security, Contiki OS

I. INTRODUCTION

In IoT network security is of great importance in real-life applications. Security in the realm of communications and networking has been the subject of much research [1]. Special attention is paid to trust management because the trust may be integrated into communication and network protocols design. In addition, the need for cooperation between participating nodes is critical in the development of trust relationships as these determine the availability, dependability, and secure operations of the network [2,3].

The trust management idea was proposed as a way to address issues like key management, authentication mechanisms, and safe routing. The open problem to solve is to come up with an efficient trust computation method for an IoT device acting as a service requester to dynamically assess the service trustworthiness of another IoT device acting as a service provider while taking into account the target IoT device's service history and its social relationships with other IoT devices [4]. Although problems are due to the complexity, characteristics, and nature of the IoT, other problems arise. The latter is due to the wrong understanding of the need for trust management, as well as to the complexity of the trust management system itself. The most relevant issues that can disrupt trust are interoperability and dynamicity [5]. The multiservice approach is one significant feature of IoT that has not been explored in trust systems for sensor networks [1,6,7]. There are heterogeneous nodes in IoT ecosystems that can provide various types of services [1,4,7]. Each service necessitates a unique set of resources from each node. Indeed, any computer may connect or detach from the network at any moment, causing a variety of network conditions to change and communication to be disrupted. In these circumstances, the trust management system must respond to any changes in the environment by updating itself [5,8].

ON-OFF attacks (OOAs): A good and bad service is respectively provided by the malicious node. The objective is to maintain its integrity and to undermine the network by offering good advice for malicious nodes and bad advice for trustworthy nodes [1,2,5,9]. This attack also takes advantage of the complex properties of confidence by contradictory behaviors. OOAs are, on the other hand, a selective form of attacks [1,2,7,9].

The malicious devices will randomly provide good and bad services to avoid being classified as a node of low trust. An OOA attacker often has different characteristics to deal with various neighbors to gain incompatible opinions of trust of the same node. The use of conventional trust management systems is difficult to detect this type of attack [2,7].

The goal of this study is to enhance trust management in the IoT environment by detecting OOAs based on the behavior of this kind of attack: Status (ON-OFF) and inconsistent behavior of each node. One is based on Artificial Neural Networks (ANN) that assists the network in detecting any inconsistent behavior. And another, to analyzing the status (ON-OFF) and the time for each node that is being exploited by the attacks.

II. RELATED WORKS

Several trust management systems have been suggested for wireless networks, such as ad hoc and sensor networks, etc... [1]. Recently, some trust management research has been conducted in the IoT. As we got that managing trust in the Internet of Things is still an open issue for researchers [1-9] they have addressed many problems related to the IoT. One of these problems is related to the attacks.

The researchers divided the attacks in the IoT into Self-promotion attacks (SPA), Bad-mouthing attacks (BMA), Ballot-stuffing attacks (BSA), Opportunistic service attacks (OSA), and OOAs [5,8,10]. In SPA attacks, the malicious node

manipulates its reputation by providing good recommendations for itself [5,8]. In BMA attacks the malicious node manipulates the reputation of another trusted node by providing bad recommendations for it [8]. BSA attacks are also known as Good-mouthing attacks. In these attacks, some malicious nodes can cooperate to trigger the attack. One malicious node manipulates the reputation of another malicious node by providing good recommendations for it [5,8,10]. OSA attacks the malicious node attempts to become opportunistic by providing a good service to keep its reputation high. The aim is to cheat with other malicious nodes to carry out bad mouthing and good mouthing attacks [8,10].

Since the limits of trust management approaches have been studied, the OOAs are of special significance. because it alternately, masquerading as a good node (trusted node) and a bad node (untrusted node), this attack will lead the system to interpret a bad node's behavior as a momentary mistake. After that, the rogue node is activated and no longer visible on the network and growth fast. The OOAs are present in two statuses: The ON status is considered an attack status (active node), whereas the OFF status is considered normal (inactive node). The attacker alternates between these two statuses. In a recent study [9] this newly introduced reward and punishment system was discovered to be the only way to protect against OOAs in IoT environments. Accordingly, on a per-node basis, it calculates the trust value and is extracted from direct observations to provide the service relevant to nodes. This technique takes a lot of time and consumes resources.

Our paper is based on the authors in [11] who used machine learning to detect some attacks in IoT such as the Blockholes Attack by helping the Intrusion Detection System (IDS). Also, we depended on the survey on machine learning-based intrusion detection approaches [12]. In addition to that, the authors in [13] proposed a gateway-based trust management system and an algorithm for the computation of trust for the devices. Our method relies on ANN to help us analyze the behavior of OOAs based on the dataset that was generated via the Contiki OS simulator and then classified into normal and malicious nodes. Table I presents the type of attacks in the IoT and the kind of method which is used to solve them.

Table I. Types of attacks in IoT and used methods

Studies and attacks type	**Used Method**
SPA [8]	Direct service quality trust assessment and feedback propagation.
BMA [8,10]	Credibility to rate a recommender.
BSA [4,8]	Honesty trust assessment and feedback propagation.
OSA [8,10]	Adaptive to adjust the weights of direct and indirect service quality trust dynamically
OOA [9]	Reward and punishment scheme
OOA Our Method	Artificial Neural Network Multi-Layer Perceptron (**ANNMLP**)

III. THE SCENARIO OF ON-OFF ATTACKS

The support multiservice approach is an important feature of the IoT. OOAs are regarded as a type of selective attack. Malicious nodes that perform actions based on the type of service they provide to other nodes in the network can target multiservice IoT architectures. To avoid being classified as an untrust node, malicious devices may provide both good and bad services at random. In order to achieve inconsistent trust opinions of the same node, an OOA attacker might act differently with various neighbors. Traditional trust management methods have a hard time detecting this type of attack. In the OOAs, malicious nodes stop providing services that are offered on the network. This attack exploits the dynamic properties of trust through time-domain and inconsistent behaviors. OOAs are exploited between nodes by two gaps in the network.

A. Through Time

OOAs have two statuses: The ON status is considered the attack status while the OFF status is a normal state and the node behaves like a good node. The attacker switches between these two statuses. When a node (j) is still ON status (active node not in idle status). It maybe brings any malicious node. Like, Fig. 1 scenario, node 2 fetch malicious node 4 between multiservice.

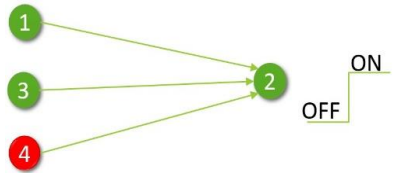


Fig.1 OOAs scenario

B. Inconsistent Behaviours

By behaving as a good node and as a bad node alternately, sending a good node (trusted node) and another bad node. In other words, the malicious nodes alternately behave well and badly by sending trusted data and untrusted data, which makes the node's ability to receive data in doubt.

IV. METHODOLOGY

A. *Proposed Model to Detect ON-OFF Attacks*

Our paper aims to enhance trust management in the Internet of things to detect OOAs based on the features that exploit the network through it under multi-service conditions. A Multilayer Perceptron (MLP) neural network is applied in the research work for detecting attacks in computing fields. In our paper, we initialized a real dataset based on the OOAs' behavior. According to the described OOAs above, In the network area, by using Contiki OS, we initialized a real network to capture the statuses of each node (ON-OFF status) and, at the same time, capture the radio message parameters for each node to manipulate inconsistent behavior. Our proposed model consists of three phases to detect attacks as we describe in Fig. 2.

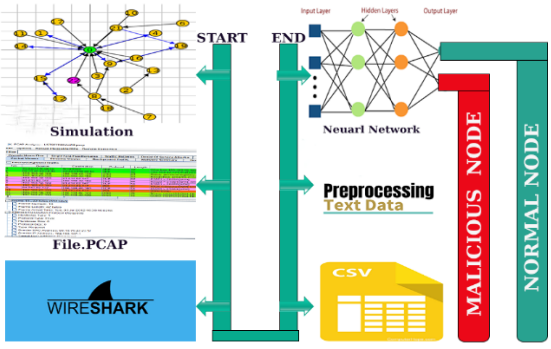


Fig.2 ON-OFF attacks model

- Simulation Phase: The open-source Contiki/Cooja simulator is used in the simulation process to produce data packet information that is similar to real-time data packets. For further processing, the packet capture file (.pcap) developed by the Cooja simulator is first converted to the CSV file format.
- Pre-processing Phase: Pre-processing stages such as feature extraction and normalization will be performed on the CSV files. In this step, the dataset will be able to feed into the MLP network after this phase is completed.
- ANN Phase: In this phase, the pre-processed datasets are produced, which consists of a mixture of radio messages and ON-OFF status for each node. Then the dataset is fed into an ANN system. The input, weight, and bias values are adjusted. The input represents the features for each node based on OOAs and how it works. The initial weight for each node is zero.

ANNMLP is a convenient way to discover any inconsistent change by learning from the recurring characteristics of each node. Then it classifies the data based on the features of the node, either a normal or malicious node. Also, one important thing is the status of the nodes that OOAs exploit the time when the node is still ON. It could bring any malicious or untrustworthy node. The features of node status consist of a status node ON, a timeline, and the log-output for each node. Additionally, there are extra features. Through this, the ANN can monitor and classify the node statuses. We discarded the status of nodes when they were OFF-status because the node in idle mode doesn't affect any node maliciously. Equation 1 represents the status calculation for each node.

$$SE_j = \sum_{x=0}^{k} WT_{status}^{ON} \quad (1)$$

SE_j represents the statuses ON for node j where j, T, k are the number of nodes, the timeline for each status ON and the number of ON status per node, respectively. W represents the adjacent factor status initialized by zero.

In our model, just we focused on the features through which OOAs exploit the network, like information of radio messages that consists of time, information of node, node number, type of protocol, source, and destination. Equation 2 represents the trusted equation calculation between nodes.

$$TE_j = \sum_{j=0}^{N} w_i t_i \quad (2)$$

TE_j represents the trusted equation between nodes and t_i, w_i represents the trusted value, adjacent factor between nodes respectively. Through these parameters, the ANN itself can detect any inconsistent behavior by learning from repeated data, then it classifies the normal or malicious nodes. The parameters SE_j and TE_j will be entered into the ANN as features for node j. Equation 3 represents the trusted calculation for node j between nodes.

$$TC_j = TE_{j-1} + SE_j \quad (3)$$

Our proposed model consists of three main phases as we described in Fig. 2. Additionally, Application program interface (API). The input for our model IoT simulation dataset generated by Contiki-Cooja simulator and the output ANNMLP Model for classifying attacks.

In simulation phases, we initialized 20 sky motes one is the Sink node and which represents the root and the others are senders. We captured the .pcap file using a 6LoWPAN packet analyzer. In this phase, we monitored and analyzed all the data that come from each node especially, the data which we need to detect OOAs attacks based on the time and the statuses for each node, and through which the node will be exposed to any attack. Furthermore, other features like the source of the node to know the source of attacks and break it.

We utilized a Machine Learning Studio (MLS) in our paper since it provides a versatile and configurable framework for ML. Each step of this procedure is carried out by a different sort of module, which may be changed, added to, or removed without affecting the rest of our experiment. In Pre-processing Phase, the CSV files will undergo the pre-processing stages then the data ready to fitting into ANNMLP. We split the data into 70 % training data and 30 % testing data throughout the neural network phase. The training data comes from the features and parameters of each node, which are put into the learned model, and then into the MLP network, which classifies the nodes as normal or malicious.

The score A machine learning model, also known as prediction, is the method of producing values from new input data using a trained machine learning model. Model evaluation, evaluates a scored classification or regression model using standard metrics. This model, we can learn the accuracy, precision, recall, and F1 score for our model. Fig. 3 shows the full proposed model and flowchart for our model.

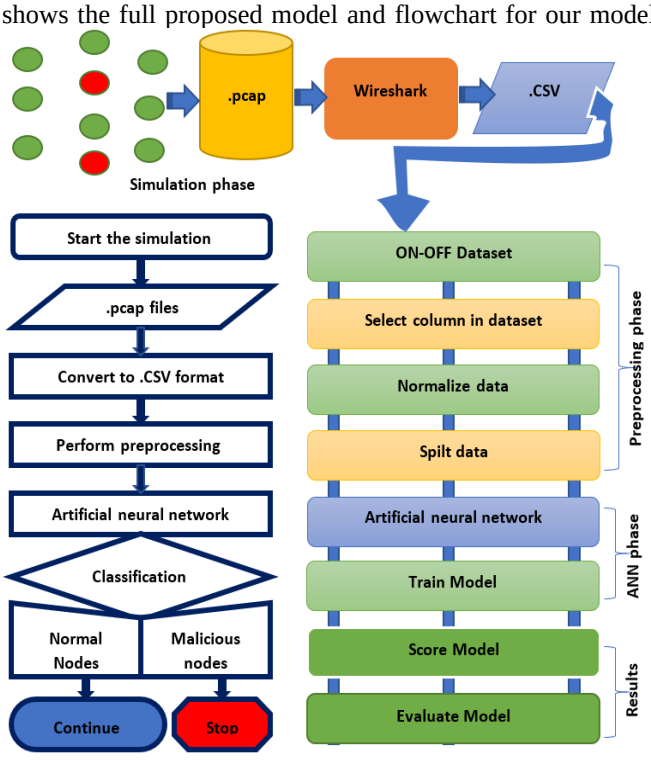


Fig.3 Proposed model and flowchart

B. *System Implementation*

In the COOJA simulation framework included in Contiki-OS, we introduced trust management schemes and built our simulation environments. Table II shows the network area configurations in Contiki OS.

TableI Network configuration

Network Settings
Simulator: Cooja under Contiki 7.2 OS
Radio environment: Unit disk graph medium (UDGM)
Network area: 180 m × 180 m
Type: Sky mote
Number of nodes: 20 senders and 1 sink
Range of nodes: Trans. range: 50 m
Physical layer: IEEE 802.15.4
MAC layer: IPv6
Network layer: RPL
Transport layer: UDP
Simulation duration: 5 h

Contiki MAC is a task cycle system that allows nodes to save energy by turning off their radio for the majority of the time while still relaying multi-hop messages.

We initialized 20 motes as a sender and one mote as a root. Three of these nodes are vulnerable to any risk, not a malicious node but maybe bring any malicious node because the status of these nodes are active all time. Two nodes work as malicious nodes while the other nodes are normal. After configuring the network, the window shown in Fig. 4. appears.

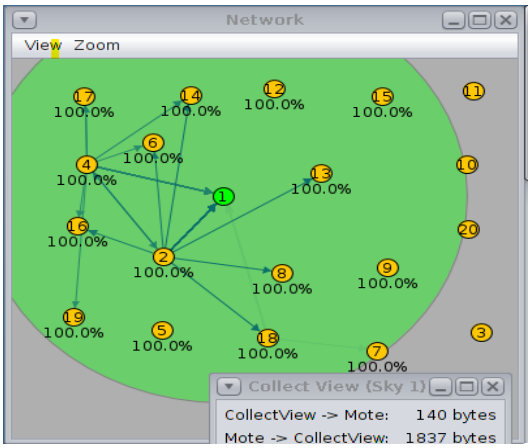


Fig.4 Network Area

Nodes 4,8,12 are vulnerable to any risk because the status ON for these nodes is active all the time. In other meaning these nodes send and receive data. Nodes 9 and 10 are malicious, whereas other nodes are normal. To apply the behavior of OOAs we wrote simple code to ensure nodes 4,8,12 are in active mode all the time by using the time delay function as shown in Fig. 5.

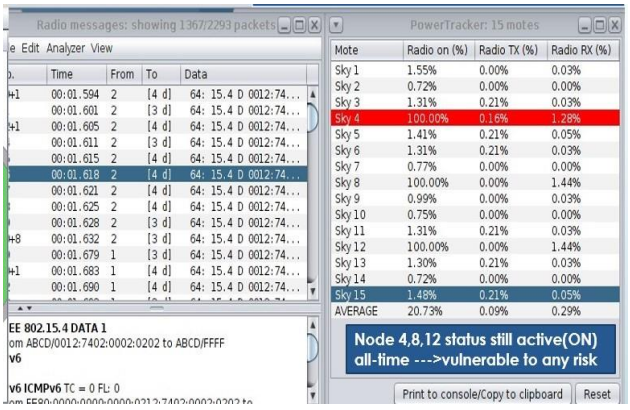


Fig.5 Radio messages and statuses

Based on the behavior of OOAs which works as bad and good behavior alternately we considered the malformed packet as a malicious node because the nodes that received the malformed packet one-time receive and resembled the packet, so we considered trusted nodes, and another time it can't resemble the packet also we considered untrusted nodes. Therefore, these are the reasons for our choice of the malformed packet which plays a scenario OOAs (inconsistent behavior). Fig. 6. Illustrates the events for malicious nodes 9 and 10 respectively.

Fig.6 Malicious nodes events

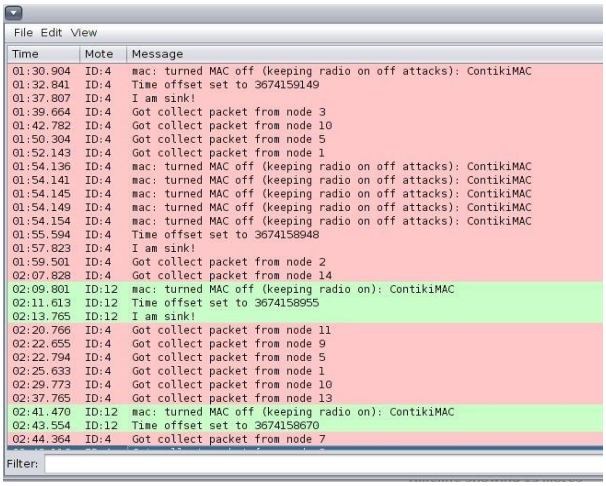
Fig. 7. represents the real implementation of Fig. 3 for our model in MMLS.

Fig.7 MMLS implementation

V. Results and Discussion

Fig. 8. shows the output nodes where nodes 4,8,12 in danger mode which increases the possibility of receiving data from

malicious nodes such as node 9 and 10, and this leads to its spread in the network quickly, and it is injected with OOAs.



File Edit View

Time	Mote	Message
01:30.904	ID:4	mac: turned MAC off (keeping radio on off attacks): ContikiMAC
01:32.841	ID:4	Time offset set to 3674159149
01:37.807	ID:4	I am sink!
01:39.664	ID:4	Got collect packet from node 3
01:42.782	ID:4	Got collect packet from node 10
01:50.304	ID:4	Got collect packet from node 5
01:52.143	ID:4	Got collect packet from node 1
01:54.136	ID:4	mac: turned MAC off (keeping radio on off attacks): ContikiMAC
01:54.141	ID:4	mac: turned MAC off (keeping radio on off attacks): ContikiMAC
01:54.145	ID:4	mac: turned MAC off (keeping radio on off attacks): ContikiMAC
01:54.149	ID:4	mac: turned MAC off (keeping radio on off attacks): ContikiMAC
01:54.154	ID:4	mac: turned MAC off (keeping radio on off attacks): ContikiMAC
01:55.594	ID:4	Time offset set to 3674158948
01:57.823	ID:4	I am sink!
01:59.501	ID:4	Got collect packet from node 2
02:07.828	ID:4	Got collect packet from node 14
02:09.801	ID:12	mac: turned MAC off (keeping radio on): ContikiMAC
02:11.613	ID:12	Time offset set to 3674158955
02:13.765	ID:12	I am sink!
02:20.766	ID:4	Got collect packet from node 11
02:22.655	ID:4	Got collect packet from node 9
02:22.794	ID:4	Got collect packet from node 5
02:25.633	ID:4	Got collect packet from node 1
02:29.773	ID:4	Got collect packet from node 10
02:37.765	ID:4	Got collect packet from node 13
02:41.470	ID:12	mac: turned MAC off (keeping radio on): ContikiMAC
02:43.554	ID:12	Time offset set to 3674158670
02:44.364	ID:4	Got collect packet from node 7

Filter:

Fig.8 Node Output

After capturing and analysing nodes, the data shows the time which OOAs exploit the network through it when nodes in an active this means that node 4,8,12 take more time compared to others as in Fig. 9.

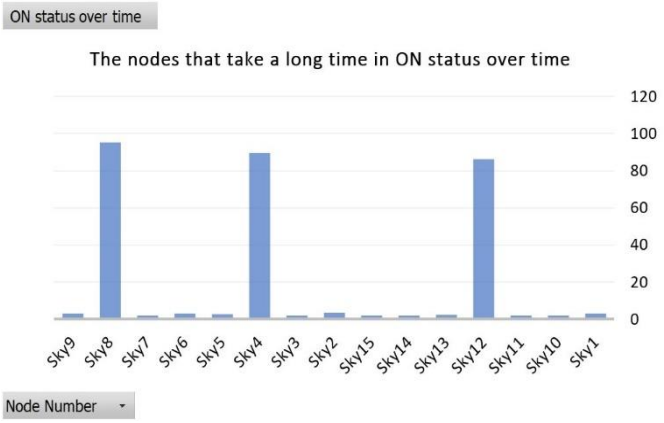


Fig.9 Radio ON over time

Fig. 10. clarifies the real dataset captured by Contiki OS with the help of 6lowpan protocol based on the behavior of OOAs. This data was visualized before entering into an ANN.

ON_OFF_Attacks › ONOFFATTACKS.csv › dataset

rows 2105 columns 16

	Time	From	To	Data	No.	Time (2)	Source	Destination	Protocol	Length	Info	Output	NodeNumber	Radio_ON	Radio_TX	Radio_RX
view as																
	1533	2	4_8_12	64: 15.4 D 0012:7402:0002:0202 FFFF|IPv6|ICMPv6 RPL DIS|0000	1	0	fe80::212:7402:2:202	ff02::1a	ICMPv6	66	RPL Control (DODAG Information Solicitation)	0	1	0.007	0	0.0016
	1537	2	4_6_8_12	64: 15.4 D 0012:7402:0002:0202 FFFF|IPv6|ICMPv6 RPL DIS|0000	2	0	fe80::212:7402:2:202	ff02::1a	ICMPv6	66	RPL Control (DODAG Information Solicitation)	0	2	0.0714	0.0506	0
	1540	2	4_8_12	64: 15.4 D 0012:7402:0002:0202 FFFF|IPv6|ICMPv6 RPL DIS|0000	3	0.08	fe80::212:7402:2:202	ff02::1a	ICMPv6	66	RPL Control (DODAG Information Solicitation)	0	3	0.0046	0	0.0017
	1543	2	1_4_8_12	64: 15.4 D 0012:7402:0002:0202 FFFF|IPv6|ICMPv6	4	0.081	fe80::212:7402:2:202	ff02::1a	ICMPv6	66	RPL Control (DODAG	0	4	0.587	0	0.0506

Fig.10 Dataset before entering into an ANN

ON_OFF_Attacks › Score Model › Scored dataset

rows 631 columns 18

2.238372	9	4_8_12	00000000 00000001 040E0008 0C0A0700 01000001 00FFFFFF 081E4040 00000000 00000000 00000000 AAAA0000 00000000 00000000 00000000	1.681859	1.520659	fe80::212:7409:9:909	ff02::1a	ICMPv6	0.331141	RPL Control (DODAG Information Object)	1	0.474484	-0.517279	-0.482835	-0.4625	1	0.999335
-0.832853	10	4_8_12	97: 15.4 D 0012:740A:000A:0A0A FFFF|IPHC|ICMPv6 RPL DIO|AAAA0000 00000000 00000000 00000001 040E0008 0C0A0700 01000001 00FFFFFF 081E4040 00000000 00000000 00000000 AAAA0000 00000000 00000000 00000000	-1.13221	-0.725065	fe80::212:740a:a:a0a	ff02::1a	ICMPv6	0.331141	RPL Control (DODAG Information Object)	1	0.242477	-0.511915	-0.409311	-0.450633	1	0.992817
0.296176	2	4_8_12	97: 15.4 D 0012:7402:0002:0202 FFFF|IPHC|ICMPv6 RPL DIO|AAAA0000 00000000 00000001 040E0008 0C0A0700 01000001 00FFFFFF 081E4040 00000000 00000000 00000000 AAAA0000 00000000 00000000 00000000	0.625349	0.519346	fe80::212:7402:2:202	ff02::1a	ICMPv6	0.331141	RPL Control (DODAG Information Object)	0	-0.221536	-0.513892	-0.419815	-0.458544	0	0.000001
-0.910309	13	4_8_10_12	64: 15.4 D 0012:740D:000D:0D0D FFFF|IPv6|ICMPv6 RPL DIS|0000	-1.393869	-1.453433	fe80::212:740d:d:d0d	ff02::1a	ICMPv6	-1.650141	RPL Control (DODAG Information Solicitation)	0	1.170504	-0.517279	-0.482835	-0.4625	0	0
-0.928399	1	4_8_12	64: 15.4 D 0012:7401:0001:0101	-1.639072	-1.932009	fe80::212:7401:1:101	ff02::1a	ICMPv6	-1.650141	RPL Control (DODAG	0	1.40251	-0.501752	-0.776928	-0.357678	0	0.000017

Fig.11 Dataset after entering into an ANNMLP

If we observed nodes 4,8,12 connected to node 9 another time connected to node 10 and the output (scored label) is one this means nodes 4,8,12 bring the malicious nodes this leads to its rapid spread while nodes 2,13,1 connected to nodes 4,8,12 and the output is zero this means there are no malicious nodes. The important note after nodes connected 9 and 10 to nodes 4,8,12 and became malicious nodes also nodes 2,13,1 became malicious nodes because it connects to nodes 4,8,12, etc... Therefore, OOAs are dangerous attacks it grows quickly in the network area through good and bad dealings. Fig. 11. illustrate the dataset after analyzing data and score labels.

The bold line in Fig. 12. represents the performance result.

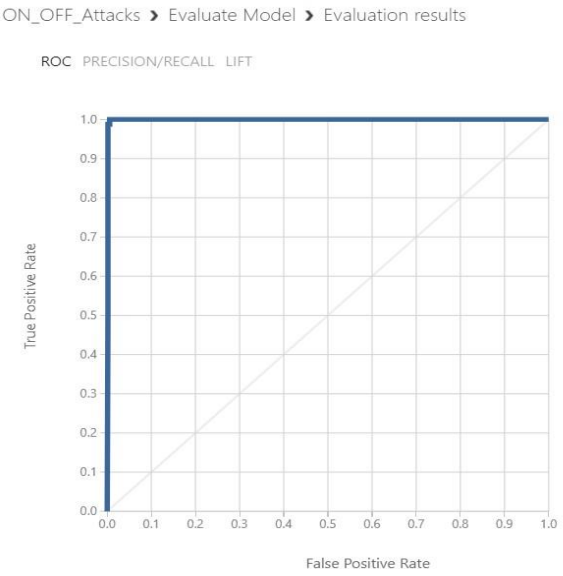


Fig.12 Model performance

In our model, the accuracy, precision, recall, and F1 score represents 99%,98%,98% and 98% respectively as in Fig. 13.

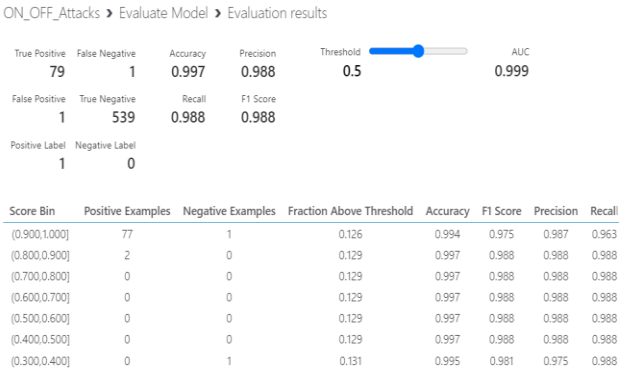


Fig.13 Model parameters performance

In MMLS introduces the API as a web service we can bring our model by using the API key to verify from the nodes either if it is normal or malicious nodes based on the features or parameters that are coming from nodes the API shows in Fig. 14.

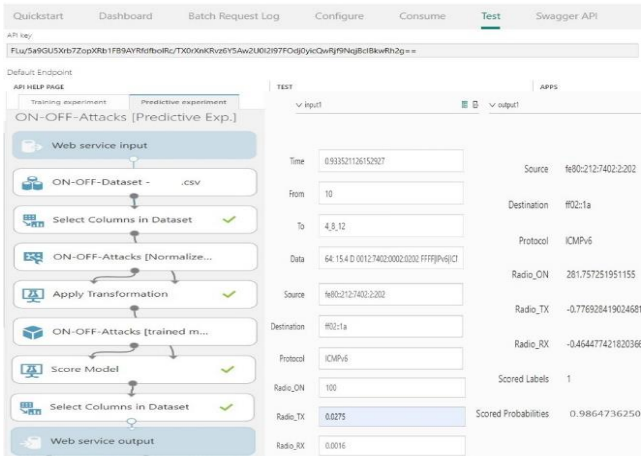


Fig.14 Application program interface

VI. CONCLUSION

We proposed a model to enhance trusted management in IoT to detect OOAs. By creating a real dataset using the Contiki OS simulator based on the behavior of OOAs that exploit the network through it. Then we analyzed and monitored the data by using an ANN to classify the node as a malicious or normal node. Finally, we implemented our method by using MMLS Additionally, we created a web service to verify new nodes' behaviors and classify them quickly into normal and malicious nodes. Also, we can bring our model through an API key to verify from new nodes, either the nodes with OOAs or not.

REFERENCES

[1] J. Henrique, “Mitigating On-Off attacks in the internet of things,” Inter. J. Dis. Sensor Networks, pp.1-8, DOI.10.1155/2015/859731, November 2015.

[2] A.V., P. Zhang, “Trust management for internet of things,” J. Net. Comp., pp.120–134, Appl. (42), 2014.

[3] Ray, S. K., Gutierrez, Airehrour, “Secure routing for the internet of things,” A survey. J. Net. Computer App., 66,198-213, 2016.

[4] R. Giraud, L. Atzori, M. N., “Trustworthiness management in the social internet of things,” IEEE Transl. Data Manage, 26 , pp. 1253–1266, May 2014.

[5] Y. Saied, D. Zegh., M. Laurent, A. Oli., “Trust management system design for the internet of things,” context-aware and multi-service appro., 39, pp. 351–365, 2013.

[6] A. Shenfield, D. Day, A. Ayesh, “Intelligent IDS using ANN,” ICT Express,pp.95.99,DOI:https://doi.org/10.1016/j.icte.2018.04.003,2018

[7] Abderrahim, & Elhedhili, “Trust management system mitigating On-Off attacks and dishonest recommendations for the Internet of Things,” *DTMS-IoT*, 2016.

[8] F. Bao, I.R. Chen, J. Guo, “Adaptive and survivable trust management for the community of interest-based internet of things systems,”11th Inter.Symp. on Auto. Decent.Sys., Mexico,2013.

[9] C. Mendoza, J. Mendoza, “mitigating on-off attacks in the internet of things using a distributed trust management scheme,” Inter. J. Dis. Sens. Net., Article 859731(11), 2015.

[10] D. Chen, G. Chang, D. Sun, X. Wang, “Trust management model based on a fuzzy reputation for the IoT,” TRM-IoT, Comput. Sci. Inf. Syst., pp.1207–1228, April 2011.

[11] W. Khreich, B. Khosravifar, A. Hamou-Lhadj, and C. Delhi, “An anomaly detection system based on variable N-gram features and one-class SVM,” Info. and Sof. Techno., vol. 91, pp. 186–197, 2017.

[12] K.Costa, J. Papa, C.Lisboa, “Survey on machine learning-based intrusion detection approaches,” IoT ,Compu. Net., PII: S1389-1286(18)308739, DOI:https://doi.org/,10.1016/j.comnet.2019.01.023, 2019.

[13] M. Apte,S. Kelkar, A. Dorge ,S. Deshpande. Bomble ,A. Dhamankar, “Gateway based Trust Management System for Internet of Things,” ISSN: 2237-0722, Vol. 11 No. 4 (2021), August 2021.

[14] Contiki Operating System for the Internet of Things, Online Available: HTTP:// www.contiki-os.org/.

[15] Azure Microsoft for ML, Online Available: https://studio.azureml.net/.