EAI Endorsed Transactions
on Internet of Things
Research Article EAI.EU

Detections of IoT Attacks via Machine Learning-Based Approaches with Cooja

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

[1]Department of Computer Engineering at Kocaeli University, Kocaeli, Turkey
[2]Department of Software Engineering at Kocaeli University, Kocaeli, Turkey

Abstract

Once hardware becomes "intelligent", it is vulnerable to threats. Therefore, IoT ecosystems are susceptible to a variety of attacks and are considered challenging due to heterogeneity and dynamic ecosystem. In this study, we proposed a method for detecting IoT attacks that are based on ML-based approaches that release the final decision to detect IoT attacks. However, we have implemented three attacks as a sample in the IoT via Contiki OS to generate a real dataset of IoT-based features containing a mix of data from malicious nodes and normal nodes in the IoT network to be utilized in the ML-based models. As a result, the multiclass random decision forest ML-based model achieved 98.9% overall accuracy in detecting IoT attacks for the real novel dataset compared to the decision tree jungle, decision forest tree regression, and boosted decision tree regression, which achieved 87.7%, 93.2%, and 87.1%, respectively. Thus, the decision tree-based approach efficiently manipulates and analyzes the KoÜ-6LoWPAN-IoT dataset, generated via the Cooja simulator, to detect inconsistent behavior and classify malicious activities.

Keywords: IoT security, Attacks, Machine Learning-based approaches, Decision tree-based models, Cooja simulator.

Received on 01 March 2022, accepted on 02 April 2022, published on 07 April 2022

doi: 10.4108/eetiot.v7i28.324

1. Introduction

An Internet of Things (IoT) is a network of physical objects containing sensors, actuators, microcontrollers, and smart appliances that gather and transfer information and interact with their surroundings [1], [2], allowing these devices to generate and exchange data with minimal human intervention. It is one of the most promising technologies and the world is already beginning to utilize various IoT technologies. It communicates with each other via various protocols [3] as well as interacts with a wide range of applications, including smart cities, building automation, safety, surveillance systems, logistics, healthcare, economy, calamity and agriculture [4], [5], [3]. Therefore, it offers a large number of attractive qualities that have made us rely on it in our daily applications with best-effort and real-time [6], [7].

The IoT cloud provides capabilities for collecting, processing, managing, and storing massive amounts of data in real-time [8], [9]. This data may be easily accessed remotely via industries, governments, monitoring tools, and related services, allowing them to make decisions as needed [10], [11]. It is essentially a powerful, high-performance network of servers designed to do high-speed data processing for billions of connected devices [12].

IoT technologies have certain properties in common that are described as heterogeneity, auto-configuring, dynamic ecosystem, smart, large scale, and connectivity [4], [13], [14], [15]. For example, the IoT ecosystem includes extremely different technologies and protocols, adaptive protocols, a variety of factors that may be influenced in order to adapt to environmental changes, etc. These components (large scale) work together in a cooperative and smart way to share their collected data and services [16]. In many cases, the connected devices are required to offer secure and reliable services to an applicant [17].

*Corresponding author. Email: 195112025@kocaeli.edu.tr

The development of technologies day by day increases the various characteristics and techniques of the IoT ecosystem, therefore, raising new security concerns [18], [19] as well as vulnerabilities that cannot be fully addressed by using traditional security solution formulation.

Nowadays, the IoT is facing an increase in threats and security vulnerabilities. Current security techniques may be used to defend against specific IoT attacks. However, the traditional approaches may be inefficient in the face of technological advancements, as well as a variety of attack kinds and severity levels. Thus, it is basic and important to connect IoT and Machine Learning (ML) technologies in order to enhance their cooperation in many aspects. Therefore, enabling ML in IoT for learning and analyzing the behaviors of IoT devices/objects, and systems based on prior information and experiences may allow the IoT ecosystem to effectively manage the unexpected deterioration that is frequently caused by anomalous conditions. Therefore, ML methods have seen significant technical development, opening up numerous new research directions to solve current and future problems in various sciences [20], [21].

The IoT is a master plan that intends to interconnect things to the Internet in order to increase their usefulness [19]. It was necessary to discover a way to integrate the IEEE 802.15.4 protocol for Low Power Wireless Personal Area Networks (LoWPANs) with the IPv6 network protocol, which has a huge address space and will allow a billion devices to connect to the internet. The invention of 6LoWPAN technology was a suitable answer to this problem [19], [18], allowing the IoT concept to become a reality. However, this was merely the beginning of a series of problems and issues, such as security [18]. 6LoWPAN is susceptible to a range of attacks that exhaust node resources and damage the network due to its inability to provide security measures [18], [22].

For the next-generation IoT systems, a powerful, dynamically improved, and up-to-date security solution is necessary. In this paper, we utilized smart technologies (ML) to find security solutions for smart environments (IoT) that make them more secure and reliable. The rapid growth of IoT exposes them to many issues and threats. ML approaches are being used as a strong technique to detect and classify inconsistent, abnormal, and harmful actions and detect incorrect IoT devices that may be due to errors.

The main contribution of this study is summarized as follows:

- Proposing a method to detect IoT attacks that relies on ML-based approaches.
- Implementing three different IoT attacks, Cooja simulator-based. The attacks are denial-of-service attacks (DoS), black hole attacks (BHA), and ON-OFF attacks (OOA).
- Generating a novel dataset based on IoT features.
- Applying the IoT novel dataset to decision tree-based models and displaying the results.

This paper is structured as follows: Section 2 shows a preview of three IoT attacks that were implemented as samples during the simulation phase. In addition to related works. Section 3 explains an overview of the 6LoWPAN protocol stack for IoT networks. Section 4 explains the methodology for detecting attacks in the IoT, the proposed method, and its implementation. Section 5 describes the tools we use to carry out our work. Section 6 discusses the decision tree-based model and results. Section 7 summarizes the conclusion of this work.

2. Related work

6LoWPAN protocol stack is vulnerable to attack due to the IoT devices are connected to an unsecured internet, therefore providing security in the IoT is critical [23]. An attacker can capture, clone, tamper with, or even destroy LoWPAN nodes [24]. Therefore, 6LoWPAN channels are generally vulnerable to a variety of security risks. The characteristics of 6LoWPAN technologies may provide attractive services compared to their peers [25]. However, they may be more vulnerable to attacks due to heterogeneity, dynamic ecosystems, etc. Although the link-layer offers encryption [24], it may not be sufficient to guarantee security to both data and signaling packets. It may be encrypted harmful packages without detecting them. The data may be encrypted to maintain confidentiality between the endpoints [26]. But it is difficult to detect and know whether the data sent is reliable, has not been tampered with, dropped, or has been breached via malicious action. Attacks may occur in different layers with different severity.

In this study, we will preview three attacks DoS, BHA, and OOA. which may be exposed to the IoT ecosystem and makes IoT devices at a critical point. DoS is an attempt to prevent the targeted user from accessing resources. This attack may occur in RPL via UDP packet flooding [27], [29]. Therefore, the attacker node sends too many requests to the root(sink), preventing normal users from accessing it in their usual way [28], [29]. BHA, which is one of the most dangerous attacks in RPL in which the malicious node drops the packets that are received from its neighbors to forward to their destination [30], [32] may drop all the packets, which is called a complete black hole, or drop some packets, called a selective forwarding attack, and this is cleverer because it is not observed and the network topology is not affected [31], [32]. Trustworthiness in IoT devices is critical. OOA is one type of attack that affects trust in the IoT and the devices and objects don't trust each other [33], [34], [35]. OOA is a kind of selective attack (inconsistent behavior) in which the malicious node switches from malicious to normal and back again to avoid being classified as a low-trust node, allowing it to remain undiscovered while inflicting harm and making the nodes suspicious to their neighbors [34], [35].

On the topic of IoT security, there have been a variety of related studies. Researchers are still working in this field. Existing research in the literature for IoT security areas offers numerous security approaches. Many approaches have been developed for detecting IoT attacks depending on solutions using traditional methods to detect specific attacks. The authors in [29] proposed an Intrusion detection system (IDS)

mechanism to detect DoS attacks, as well as the authors in [31] proposed an IDS mechanism to detect BHA in the IoT. Also, the authors in [35] proposed IDS mechanisms to detect on-off attacks. The authors [29], [31], [35] use different methods and features to detect specific attacks. This may be effective to detect a special attack, but implementing an IDS mechanism for each attack, especially with advances in technology and increasing attack types, may be inefficient. It may consume device resources as well as prevent it from being extended to detect new attacks. Currently, many researchers try to employ ML-based approaches to solve security issues, this technique becomes an ingenious solution. The authors in [36], [37], [38] are utilization the ML methods to detect attacks. A dataset may be used by other contributors such as DS2OS dataset [39], NSL-KDD dataset [40], UNSW-NB15 [41] and Bot-IoT-2018 [42] to employ in ML models. So, according to the previous reviews, various attack datasets are employed in ML-based activities, However, using the dataset may lead to its employment in more than one research and use of the same ML techniques, which leads to similar results as well as being may unrelated to the characteristics and features of the IoT. Thus, using the features related to IoT security is better for evaluations.

Our method is different from the existing works related to IoT security in many aspects in: Implementation, simulation, attacks type, dataset generation, parameter/features, used protocols and ML techniques. In this paper, we have implemented three attacks on the IoT, and we have generated a novel dataset based on IoT features (6LoWPAN) produced via Contiki OS. This dataset is called the KoÜ-6LoWPAN-IoT dataset. Then we employed this dataset on the algorithms ML-based that depend on their decisions on the tree.

3. 6LoWPAN protocol stack

IPv6 over low power wireless personal area network (6LoWPAN) is a particular instance of a low power lossy network (LLN) that allows tiny devices with restricted resources to connect to IPv6 networks, and these devices comply with the IEEE 802.15.4 standard [18]. It supports end-to-end IPv6 connectivity, allowing it to have a direct connection to the Internet with a wide range of networks (heterogeneous devices), including tiny devices [19], [18]. Interoperability is an important consideration when selecting a wireless protocol. Interoperability implies that apps do not need to know the limits of the physical connections that transport their packets. 6LoWPAN devices can communicate with any wireless 802.15.4 devices over any other IP network connection (e.g., Ethernet or Wi-Fi), in contrast to other technologies such as ZigBee devices, which can only communicate with other ZigBee devices [43].

In the network layer, the Internet Engineering Task Force (IETF) proposed routing protocol for low-power lossy networks (RPL) is the most popular routing protocol for 6LoWPAN [44] as well as used in academia and industry. Therefore, Contiki-OS provides work on the IoT and wireless sensor networks that are constrained and operate on a 6LoWPAN protocol stack [45].

In comparison to the normal Internet stack, the 6LoWPAN stack contains an extra layer, which is known as the LoWPAN adaption layer. The adaption layer rests above the IEEE 802.15.4 layer, immediately below the network layer. The adaption layer offers header compression, fragmentation, and reassembly, as well as packet forwarding services, allowing IPv6 connections to be provided to extremely tiny devices linked to the internet [46]. Therefore, the IPv6 packets are encapsulated to be sent to the underlying link-layer via the adaption layer.

Figure 1 illustrates the structure of the 6LoWPAN stack and the used protocols. The application layer in the 6LoWPAN Stack contains lightweight protocols such as Constraint Application Protocol (CoAP) that were created for the IoT and were inspired by HTTP, assuming that UDP may be used without impedance in security (RFC 8323). CoAP over UDP's message layer supports reliable delivery, basic congestion control, and flow control. It was designed with simplicity in mind, with a minimal code footprint and a small, lightweight message size [46] As a recommendation from the developers (IETF), due to the 802.15.4 MAC/PHY frame size limits, UDP is a better fit for the 6LoWPAN stack than standard TCP at the transport layer with a large size header up to 60 bytes (RFC 8323).

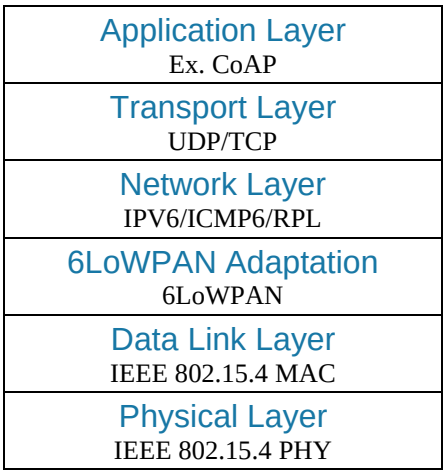

Figure 1. 6LoWPAN protocol stack

3.1. Network- IPV6 layer / Routing protocol

RPL is a routing protocol designed for low-power and lossy networks, and it has become the preferred routing protocol for IoT. It is a distance-vector routing protocol that routes the data to a destination (sink) with a short path (Optimal Path). RPL was designed to be highly adaptive to network conditions and to provide alternate routes [47] One of the main goals of RPL is to construct topologies of the network [48]. The resulting routes from a Directed Acyclic Graph (DAG) are the network topology. There is only one Destination Oriented Directed Acyclic Graph (DODAG) per root (sink) and it is the data to a root [47].

The RPL consists of four control messages that are used in the formation and maintenance of the network topology. Which are as follows: DIS, DIO, DAO, and DAO-ACK are

acronyms for DODAG Information Solicitation, DODAG Information Object, Destination Advertisement Object, and Destination Advertisement Object Acknowledgement, respectively. A node can utilize DIS to explore for DODAGs in its general vicinity. The DIO contains data that enables a node to find an RPL instance, understand its configuration parameters, choose a DODAG parent set, and keep the DODAG up to date [47]. The DODAG in the node uses the DAO to communicate destination information upward. The DAO message is unicast by the child to the specified parent. DAO-ACK is a message sent back to the DAO sender [48]. Figure 2 shows the diagram of control messages in the RPL.

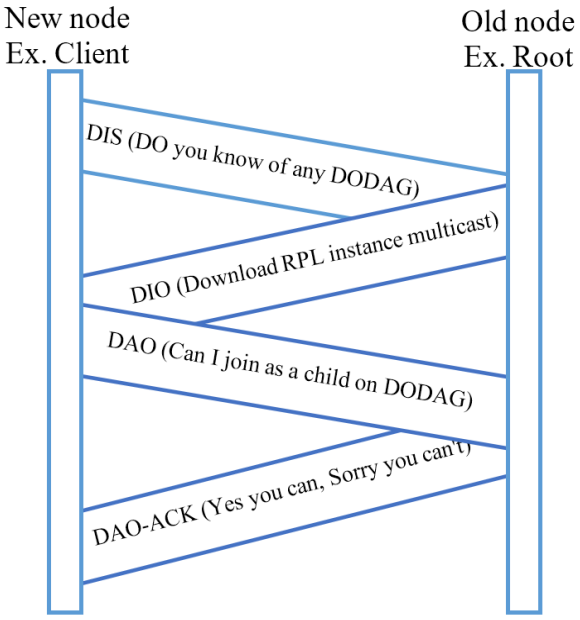

Figure 2. RPL mechanism

3.2. Internet Control Message Protocol (ICMP6)

Every IPv6 node must successfully implement ICMPv6 since it is an integral part of IPv6. The IPv6 nodes utilize ICMPv6 to report packet processing errors and conduct additional internet-layer operations, including diagnostics, such as ICMPv6 "pinging" and multicast member reporting (RFC 1885).

4. Methodology

This study depends entirely on simulation to detect attacks in IoT in all its stages to obtain the results starting from the simulation stage to reaching the stage of results. In this study, we divided our method into phases thus, our model consists of the Simulation phase, Dataset collection, and manipulation, Pre-processing phase, the decision tree-based phase, and the results phase. Figure 3 illustrates the proposed model for detecting the IoT attacks and implementation phases. It starts from the simulation phase until getting results in the evaluation.

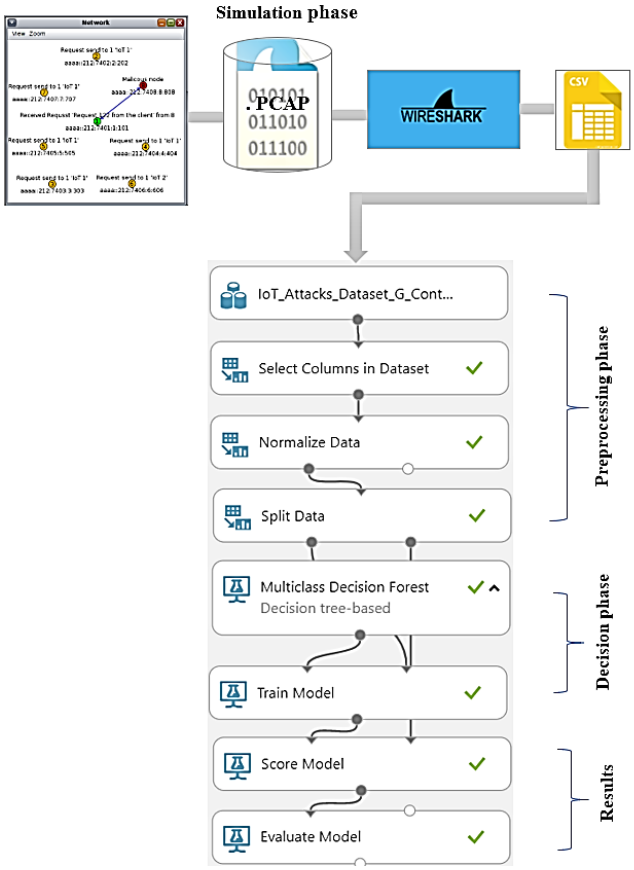

Figure 3. Proposed model for detecting the IoT attacks and implementation phases

4.1. Simulation phase:

Table 1 shows the general configuration in the Cooja network area.

Table 1. Network Configuration.

Simulator	Cooja simulator
Network area distance	190 m^2
Type of node	Sky mote
Number of nodes	7 senders and 1 root
Transport layer	UDP
Network layer	IPv6 -ICMP-RPL
Adaptation layer	6LoWPAN
Link and Physical layer	IEEE 802.15.4

In this phase, we implemented three attacks on the IoT, based on the degree of difficulty in implementation and detection. These attacks are arranged as follows: DoS, BHA, and OOA attacks. In this paper, all scenarios are configured on 6LoWPAN stacks for the IoT. Each scenario consists of eight nodes. Node 1 is the root (server/sink). Nodes 2–7 are normal nodes, while node 8 is malicious. The color for a malicious node is red. Normal and malicious nodes either request service from the root (server) or the root (sink), which

collects their data. The attacker node might be implemented on various layers. Thus, the malicious code is implemented in the transport layer as DoS, while the malicious code in BHA is implemented in the network layer and some functions are controlled in the mac layer like duty cycle to implement the OOA in addition to the transport layer. While the simulation is running in each scenario, the radio messages and the power parameters are captured. The reason is as the malicious node directly affects the power consumption, as well as the features of radio messages, change from one attack to another during the test of attacks.

As we trace the behaviors of attacks, the malicious nodes experience the most power consumption compared to normal nodes. In addition to that, the radio messages include all the matrix of 6LoWPAN protocols that we knew parameters/ features, which affect the features and parameters directly via any malicious activities. Therefore, the 6LoWPAN protocol stacks (radio messages) and the power properties are considered as criteria and inputs (dataset) to our models.

4.1.1. DoS Attacks

Denial-of-service attacks are a critical point in IoT devices due to constrained devices. So, DoS attacks are designed to make a machine or network resource unavailable to their users (clients, senders, etc.). In IoT, A DoS occurs when the attacker sends too many requests to the main server/host, making the real users of the server unable to use it. The attacker node is used to flood either traffic or requests, which causes the network traffic to overflow, preventing normal requests and traffic from entering the network. Also, the malicious node indirectly prevents other nodes from gaining access to the server. The malicious node will try to deny any normal node access to the node that is attacked. This causes the node that has been attacked to work improperly.

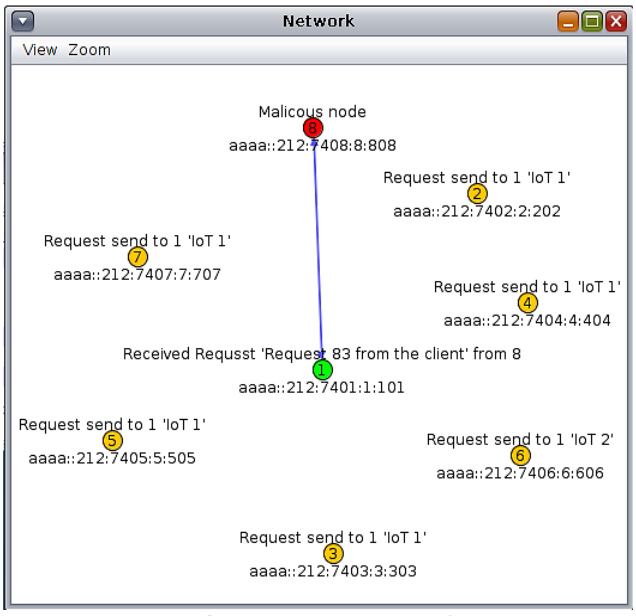

Figure 4. DoS scenario

To implement the DoS scenario, in the network area, we set up eight nodes distributed circularly as shown in Figure 4. Node 1 is a server, and nodes 2–7 are normal nodes, while node 8 is the malicious node colored in red. All the normal and malicious nodes (clients) send their requests to the server, and the server responds to them as in Figure 4. As mentioned in the first remark, the blue line between the server (node 1) and the malicious node (node 8) represents the radio traffic. The flooding of radio traffic between the server and the malicious node is obvious due to the many requests from the malicious node. In other words, the number of UDP packages produced by the malicious node is extremely high when compared to the normal nodes.

In this scenario, two periods of time are defined to send requests to the server, one for the normal nodes and another for malicious nodes. The normal nodes send requests (UDP packets) to the server every minute (normal case), while the malicious nodes send their requests to the server every second. In normal nodes, the timer sends the requests within a predefined 1-minute period to their target (server), while in the malicious node, the timer is set to send the requests within a predefined 1 second period.

Figure 5 illustrates the node's output and the requests made between clients and servers. In the output node window, it consists of three tabs: time, ID, and message. The time represents the node time events and the ID is referred to as the Node ID, whereas the message carries the request with the destination ID and the number of requests that have been requested.

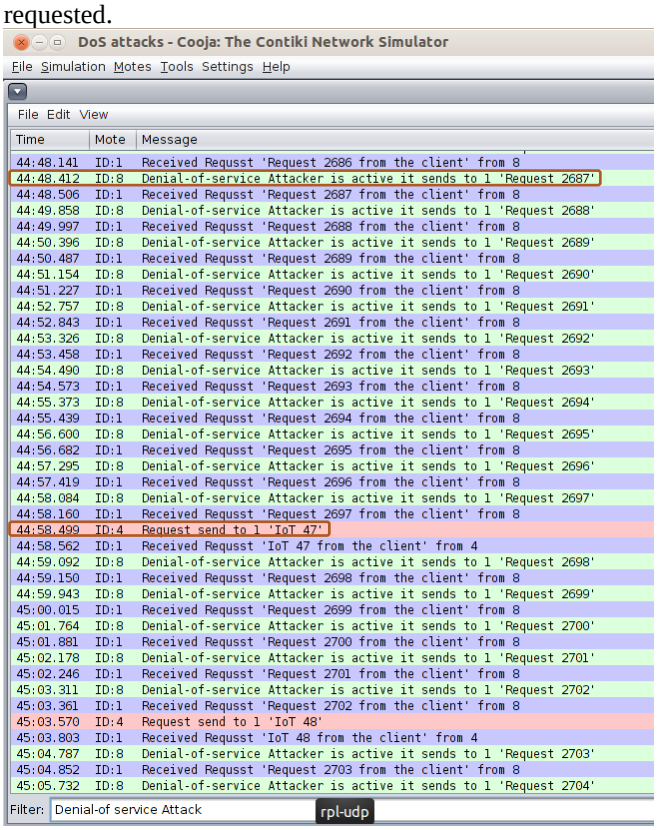

Figure 5. DoS nodes output

Where we notice that malicious node 8 sends too many requests to the server (node 1), the server may respond to it. We could observe in the second line that the DoS attacks are active and the malicious node sends a request to the server at

44 minutes and 48 seconds (44:48) and the server may receive it. Also, the malicious node sends a request to the server at the time of 44:49. The number of malicious node requests reached 2587 requests at the time 44:48, compared to normal node 4 which reached 47 requests at the time 44:58. This means the requests from malicious nodes are sent every 1 second and the process is continuous to send flooding of requests compared to the normal node. For example, node 4 sends a request to the server at 44:58 and sends it again at 45:03, which means normal nodes send their requests to the server every 1 minute, and this is normal case if there is no delay in the buffer.

4.1.2. Blackhole attacks

In a special case of black hole attacks, the malicious node drops some data packets while others are forwarded successfully. This is called selective forwarding. In another case, the malicious node does not forward any data packets. This is called a "complete black hole attack." When implementing the special case of black hole attacks, the topology of the network remained un-isolated because the malicious node is still forwarding some packets to other nodes, whereas the malicious node in complete black hole attacks must be isolated explicitly as in our scenario. In this scenario, we have implemented a complete black hole attack that isolates several nodes on the topology.

The main target in this scenario is the sink node collects data from senders whereas the senders send their packets to the sink node. In the scenario of a black hole, we set up eight nodes. Node 1 is a sink node, where node 8 is a malicious node and the other nodes are normal.

We used a multi-hop node because the black hole is more effective and affects the topology of the network. In this scenario, we placed the malicious node in a strategic position that separates several nodes that communicate with the sink via the malicious node. For further information, some nodes are located in the direct range of the sink node, while others are not, and data packets from nodes outside of radio coverage are routed through other nodes to the sink node. All data packets from senders' nodes are destined to the sink.

As appears in Figure 6, Nodes 2, 3, 4, and 5 are located within radio coverage, while nodes 6 and 7 are outside the radio area. The malicious node is the link between the nodes inside the radio area and the nodes outside the radio area of the sink node. The route of data packets from nodes 6 and 7 to the sink is passed via a malicious node. And this leads the topology of the network to isolate several nodes due to the malicious node in a strategic position as appear in Figure 7.

Each one executes and implements the malicious code in its way to examine and test its intended purpose. As well, we can implement the malicious code in different layers. In this case, we implemented and developed the malicious code in the network layer. In the malicious node of the blackhole scenario, we set some global variables (parameters) to zero to drop all packets like uip_len, uip_flags, uip_ext_len, and uip_ext_bitmap. The uip_len variable is the length of the packet in the uip_buf buffer, and the uip_buf buffer puts incoming packets in it, whereas the uip_flags variable is used for communication between the IPv6 and the application program like UDP. Therefore, many operational routers may be set to discard all packets with a hop-by-hop option header (HBH), but major difficulties still exist (RFC8200). In IPv6, we can have extension headers. If present, the HBH must be processed before forwarding the packet (Contiki team). In this scenario of black holes are present. As well as extension headers, HBH can be processed or handled by each node along a packet's delivery path until it arrives at its destination.

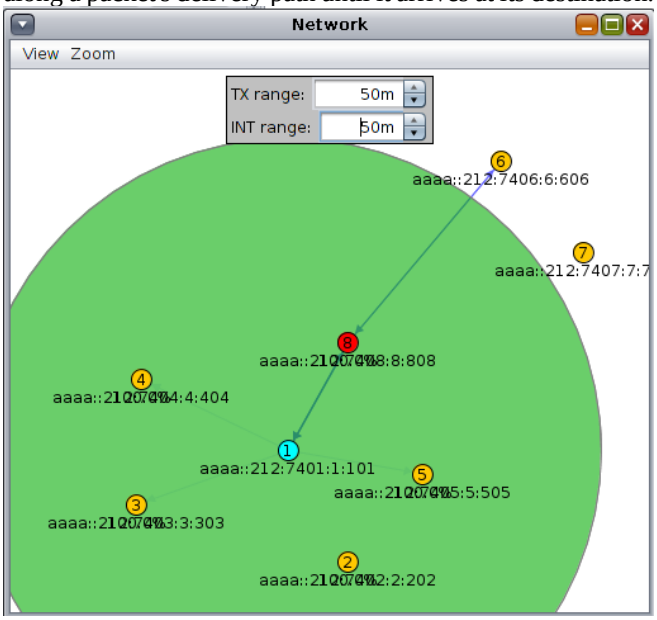

Figure 6. BHA scenario

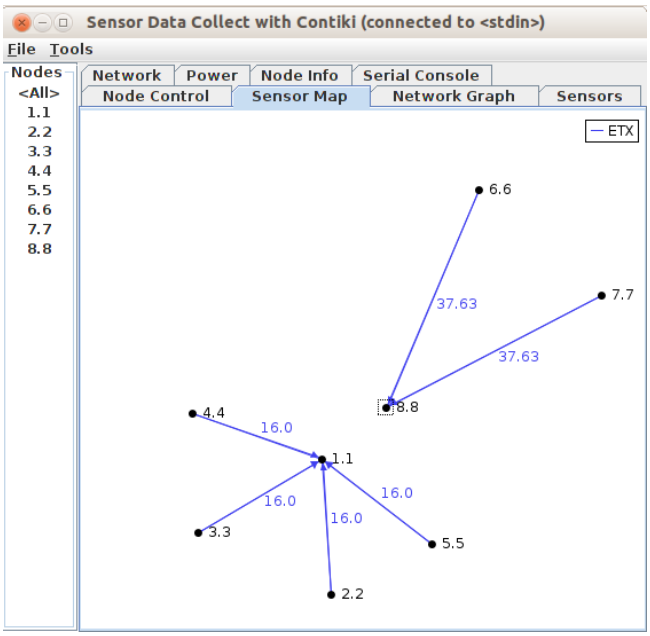

Figure 7. Effect BHA on topology

These parameters are set up and configured in the malicious node to drop all the packets by the created drop function. Thus, what's going on is that there are some conditions on the global variable. For example, if the global variable is greater than or equal to zero, it goes to the drop function. In the drop function, we return these variables to equal zero. These parameters are set up and configured in the malicious node to drop all the packets by the created drop

function. Thus, what's going on is that there are some conditions on the global variable. For example, if the global variable is greater than or equal to zero, it goes to the drop function. In the drop function, we return these variables to equal zero. Because these variables must usually be greater than zero due to the updating and changing of their values during each process. This means when these values of variables are updated or changed to any values, the malicious code returns these values to zero. Hence, the malicious node may or may not continue to receive and forward the data packets generated by other nodes. Also, the malicious node may not continue to process its generated packet. The malicious node drops the incoming and outgoing packets. The effects of the malicious node on the topology of the network are completely isolated. The global parameters in IPv6 were more effective in isolating the route between the malicious node and the destination.

4.1.3. ON-OFF attacks

OOA is a sort of selective attack to avoid it being classified as an untrusted node [34]. So, the malicious node switches its behavior from harmful to normal and back again, allowing it to remain unnoticed while launching attacks. Therefore, in this attack, there are two statuses: the ON status is called "attack," It is a critical case while the OFF status is called "normal.". This attack hits the advantage of the dynamic features of trust by exploiting the time-domain of status and inconsistent behavior [33]. The attacker swaps between ON and OFF. When the attack is ON, the malicious node initiates attacks, and when it is OFF, it does nothing [33], [35]. An OOA attacker often has to deal with various neighbors to gain incompatible opinions of trust from the same node.

ContikiMAC is a radio duty cycling technique that employs periodic wake-ups to monitor neighboring packet streams. If a packet transmission is detected during a wake-up, the receiver is kept turned ON [49] so that the packet may be received. When a packet is received correctly, the receiver sends an acknowledgement. The transceiver (transmitter-receiver) must be completely turned between OFF and ON to send and receive radio if the status is ON and save power if the status is off. Therefore, to achieve low power consumption, ContikiMAC nodes sleep most of the time and intermittently wake up to check for radio activity [49].

The purpose of this scenario is to achieve the properties of an OOA to create a real dataset that will be trained in our model to detect OOA. The malicious node in our scenario switches its action from the attacker to normal and from normal to attackers if the case malicious node's cycle is ON via sending both trusted and untrusted packages randomly.

In this scenario, we initialized eight nodes that were placed randomly in the network area as shown in Figure 8. Node 1 is a server. Node 8 is malicious, while the other nodes are normal. Node 1 is exposed to any malicious node because this node is always active (ON-status). This node wants to receive its data from neighbors' nodes in a position of trust, without any doubt. Node 8 is a malicious node that generates and sends inconsistent data alternatively. In a malicious node, the duty cycle of the ON status to OFF status is set up similarly, 50%-50% percent. This ratio makes it easier to detect malicious behavior if the radio status is ON. Normal and malicious nodes request a service from the server.

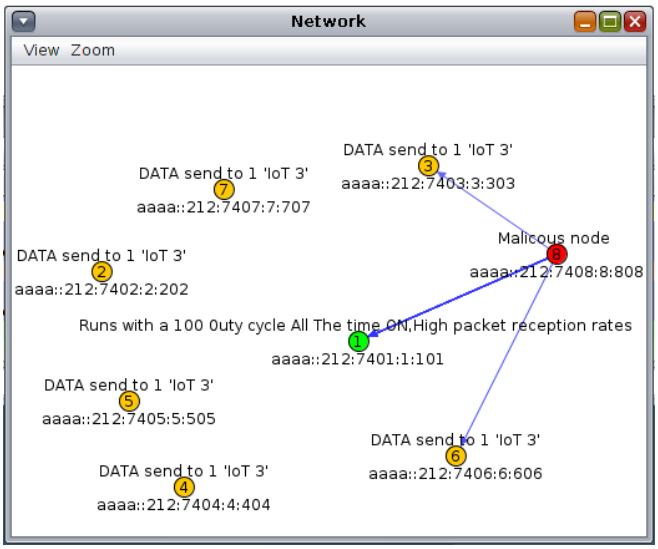

Figure 8. OOA Scenario

The malicious code was implemented only in node 8. This malicious node alternately produced trusts and untrusted packages to node 1. Node 1 was set up to have an ON status all the time. It is not known if the data being sent by the malicious node is reliable or unreliable data.

In a malicious node, we created two functions. The first function is called "Trusted_Function," and the second function is called "Untrusted_Function." The Trusted_Function sends trusted packages to node 1, while the Untrusted_Function sends untrusted packages. In another sense, the Untrusted_Function sends UDP packets that are not standard and have different parameters (i.e., payload) than their peers. Additionally, this function sends their packages at an abnormal time. For example, all the normal nodes send their UDP packets every 1 minute. The trusted function in malicious nodes also sends their UDP packets every 1 minute. Unlike the Untrusted_Function, it sends its UDP packets every 30 seconds. Thus, the malicious node did these functions, generating trusted and untrusted packages randomly to put neighboring nodes in doubt with their inconsistent behavior. Node 1 is the service provider for the other nodes. This node is vulnerable to any malicious node because it is always in the ON status (active). It is more vulnerable to attacks from malicious nodes. Because it receives huge packets from nodes, especially from malicious nodes. Node 1 may bring the malicious code from malicious nodes and spread them to other nodes.

We use the implementation of ContikiMAC as Radio Duty Cycling (RDC) in the MAC layer. ContikiMAC is a duty-cycling mechanism that allows nodes to keep their radios off as much as possible to achieve low power consumption and save energy [49]. The default radio duty cycling mechanism in Contiki 2.7 [49] uses a power-efficient wake-up mechanism with a set of timing constraints to allow devices to keep their transceivers off. By default, this setting is active when we initialize nodes in the Cooja network area. Thus, the function of the duty cycle is that we can set it up however It is desired.

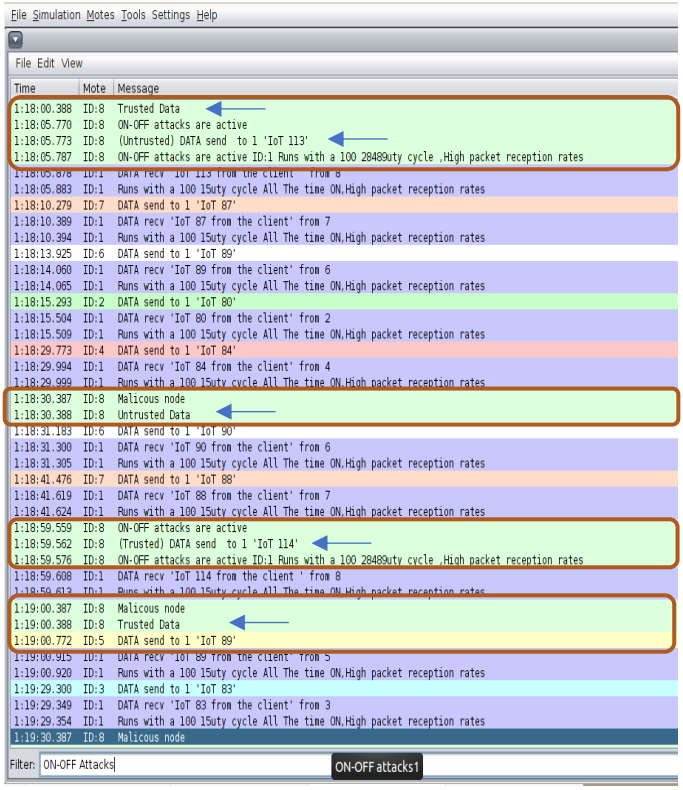

Figure 9. OOA mote output

This function was called in the OOA attacks scenario to serve the purpose of keeping the malicious node active and node 1 at risk. Alternately, the malicious node switches to lunch the malicious code (Untrusted_Function) and normal code (Trusted_Function). It remains undetected while the status is active. As for the node that is vulnerable to attack, the parameters of this function were set to 1 (100%), which means node 1 has its radio transceivers ON, high reception packets, and power consumption. The parameters for the malicious node were set to 0.5 (50%). Figure 9 illustrates the output of nodes in the OOA Scenario. This shot was captured after spending over an hour in the Cooja simulator. where observed the malicious node switch their behavior from trust to untrust randomly as clarified in the rectangle.

4.2. Dataset collection and manipulation

At this stage, we collected all the datasets generated via Contiki OS that was captured by the 6LoWPAN analyzer and Power Trace tool. Table 2 summarizes the number of observations/samples and the number of features that were collected and processed that represent the radio messages and power features in each scenario. The analysis of the radio message improves its security, and the analysis of the power improves its reliability. From all the scenarios, we obtained three datasets. The numbers of captured data are 9912, 12696, and 25072 from DoS, BHA, and OOA, respectively. These three datasets are merged into one dataset and are called the KoÜ-6LoWPAN-IoT dataset. In the process of merging the datasets at the beginning of each dataset, a sample of 20% was taken and copied one by one into the unified dataset from each attack dataset, and then the remainder was copied successively into the file that represented 80%. The datasets are not directly merged one by one. The reason is that in the ML phase when we split the data into training and testing data, the testing data was taken from the DoS sample only because the dataset was large and the testing dataset was not randomly extracted from all attack samples.

Table 2. Observations and features in each attack.

Attacks types	Simulation duration	Samples of Radio message and power trace	Number Features of Radio message/ power trace
DoS	45 min	9912	84 /12
BHA	1 hour	12696	84/12
OOA	2 hours	25072	84/12
Total	3:45	47680	96

4.3. Pre-processing phase

The practice of preparing raw data for use in a machine learning model is known as data preprocessing. It is the first and most critical step in the development of a machine learning model. Real-world data in most circumstances has noise, missing values, and is in an unsuitable format that cannot be directly used for machine learning models. Therefore, data preprocessing is a necessary step in manipulating data and preparing it for a machine learning model, as well as improving the model's accuracy and efficiency. The first thing we need to develop a machine learning model is a dataset because a machine learning model is completely dependent on data. A dataset is the collection of data for a certain topic in the appropriate format. Like the IoT attacks dataset that was generated via Contiki OS to detect malicious activity and inconsistent behavior.

The train-test split procedure is a very important part of ML. It is used to estimate the performance of machine learning algorithms. Thus, we split the dataset into 80% for training and 20% for testing. The IoT dataset contains a huge number of samples of attacks collected from DoS, BHA, and OOA attacks, which total 47680 samples of observations, 84 features from radio messages, and 12 from the power trace. The number of samples that were captured in DoS is 9912 samples over 45 minutes, while the number of samples that were captured in BHA is 12696 samples during 1 hour and the number of samples that were captured in OOA is 25072 samples over 2 hours. Therefore, the number of samples for training was 38142, while the number of tests was 9535. The features contain a mixture of categorical and numerical data.

4.4. Decision tree-based phase

At this stage, we preferred four machine learning algorithms that depend on their work and structure on a decision tree (Decision Tree-Based). The main reason for this is that these algorithms have practically demonstrated their efficiency in accurately outputting results and working on the real IoT dataset produced by Contiki OS, which is difficult for some

machine learning algorithms to deal with until these categorical data are converted into numerical data using one-hot encoding or any other method. This leads to an increase in the number of features after converting them, which leads to a slow model in efficiency.

In addition to time, in training time, the algorithms that do not work on decision tree-based models need more time to finish training than algorithms that work on decision tree-based models. For example, a neural network needs a lot of time to finish training the data on it. The AMLS takes more than an hour and 15 minutes to train the model on 47680 samples. The reason is that neural networks are more computationally expensive and do require a graphics processing unit (GPU) to finish training, unlike the algorithms that work on decision tree-based systems that are less computationally expensive and do not require a GPU to finish training. Also, when a model is requested from an external source to predict and classify new data, the response time of the model that has been trained on the decision tree-based approach is much faster compared to models that do not work on it. For example, when the random decision forest model was requested via the Postman tool, the response time of the model was most up to 6 seconds, while some models needed more than 240 seconds.

In this study, we utilized four algorithms that depend on the tree for their decisions. Two of these are for the classification of the attacks and two for regression to predict the attacks in the IoT ecosystem. To classify the attacks, we utilized multiclass random decision trees and multiclass decision tree jungle models. Also, to predict we used decision forest tree regression and boosted decision tree regression for regression. The general parameter configurations are the same in both algorithms. Thus, the number of decision trees is set at 50, and the maximum depth of the trees is 96.

5. Used Tools

5.1. Contiki OS

Contiki is a networked operating system that is designed to function on hardware with severe memory, power, and parameter constraints, with an emphasis on low-power wireless IoT devices. Contiki contains the 6LoWPAN stack network mechanisms which provide the routing protocol for low power and lossy networks IPv6 with the 6LoWPAN header compression and adaptation layer for IEEE 802.15.4 links [45].

5.1.1. Cooja simulator

Contiki includes the Cooja framework. Cooja is a powerful simulator utilized in the IoT. It is a network simulator designed for simulating sensor networks. It's a Java-based simulator that lets us write sensor nodes in the C language. [48].

5.1.2. 6LoWPAN analyzer tool

A 6LoWPAN analyzer is a tool built on the Cooja framework that captures radio messages and saves them as packets with an extension. PCAP was developed to capture all the packet data with details. Once this tool is activated when running the Cooja simulator and scenario, it does its job of capturing and saving the file with the extension PCAP automatically. This PCAP file can be opened with Wireshark to know the details of the packages closely.

5.1.3. Wireshark

Wireshark is a network packet analyzer that tries to display the packet data details [48]. It allows viewing packet data from a live network or a previously stored capture file interactively like PCAP. The PCAP format is one of the native capture file formats for Wireshark, which it can read and write. Wireshark is used to monitor network traffic and keep a close eye on what's going on in the network. It enables us to retrieve this data and convert it to CSV for further processing and analysis, as we did in our model to detect IoT attacks. By default, it displays four features of the packets, such as source, destination, protocols, and information control message. In our study, we enabled all the features and parameters for the packets, which numbered 84 features. The reason is due to the impact of attacks on some parameters in packets, which in turn make them more accurate in detecting attacks and knowing the behavior of malicious nodes than normal nodes. The features of the packet are different from one another, and the number of features is too large, thus aiding our model in detecting misbehavior in a good manner.

5.1.4. Power trace tool

Power tracing is a run-time power profiling method that estimates each node's power usage via power state tracking. by calculating the time each component spends in each power state. The ContikiMAC low power radio duty cycling mechanism is utilized by Cooja [18]. The goal of radio duty cycling is to switch the radio off as much as possible while still being able to communicate to save power. A node cannot receive transmissions from neighbors if the radio transceiver is turned off. To communicate while keeping the radio turned off as much as possible, the radio must periodically wake up between two statuses to receive packets from neighbors [18]. Nodes in a duty-cycled network do three things: transmit packets, receive packets, and periodically wake up so that it can receive packets from neighbors. These parameters that are calculated by the power traces tool are called and printed in each scenario and converted to a CSV file. Because the malicious activity in the network is affect these parameters in some types of attacks. For example, the malicious activity maybe effects on radio state and still the status ON all the time thus leading to consuming much energy and maybe vulnerability to any malicious activity. The radio must be turned completely off-duty-cycled- as possible to decrease power consumption and prevents other dangerous issues.

5.2. Azure Machine Learning Studio (AMLS)

AMLS is a platform that provides machine learning algorithms in separate modules for creating and deploying ML workflows on Azure. It is a cloud solution that helps us

speed up and manage our machine learning projects. It is a set of services and technologies aimed at assisting developers in the development and deployment of machine learning models. It may be used by machine learning experts, data scientists, and engineers in their workflows to design, train, and manage models. Individuals and teams deploying machine learning operations inside their company may use AML to move machine learning models into production in a safe and auditable environment. We can build our experiment from scratch using one of the popular programming languages such as Python or R. In addition to that, it provides ready-made modules that make it easier for us to build and test our model, which contains practical and common modules and algorithms in artificial intelligence. In this study, we utilized an MLS because it provides a flexible and extensible framework for machine learning. Each stage of this process is handled by a different type of module, which may be updated, added to, or eliminated without impacting the rest of the experiment.

6. Discussions and Results

The (ML) Decision Tree-Based Models were able to analyze and evaluate data by predicting and classifying it into normal and malicious nodes as in Figure 10. In Classification decision tree-based, the evaluation model gives us the parameter values of estimation that are derivative values from the confusion matrix. Therefore, in the multiclass random decision forest ML-based model, we obtain the overall accuracy, averaged precision, and averaged recall of 98.9%, 98%, and 97.1%, respectively. In comparison to multiclass decision tree jungle, we get 82.5%, 82%, and 44.1% for overall accuracy, averaged precision, and averaged recall, respectively. In regression decision tree-based, the metrics to evaluate the models are Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Relative Squared Error (RSE), and Coefficient of Determination (CD). Therefore, in the decision forest, tree regression is carried out at 0.138 and 0.143 for MAE and RMSE while achieving 6.75 % and 93.2 % for RSE and CD, respectively. In comparison to boosted decision tree regression, we got the MAE and RMSE of 0.12 and 0.246, while for RSE and CD we obtained 19.83% and 80.1%, respectively.

As a result, as shown in Figure 11, the multiclass random decision forest ML-based model obtained 98.9% overall accuracy in identifying IoT attacks for the real dataset IoT features-based, compared to 87.7%, 93.2%, and 87.1% for decision tree jungle, decision forest tree regression, and boosted decision tree regression, respectively. As a consequence, multiclass random decision forest produces fair results when compared to other algorithms, but this does not exclude the development of lightweight custom ML algorithms for future IoT challenges.

7. Conclusions

The cooperation of technologies among them makes them able to increase security in their aspects. The enable of ML-based techniques in finding IoT security solutions is a strong point. However, ML-based techniques make IoT devices more secure and reliable due to the heterogeneity and difficult conditions for IoT devices consequently, As summarized in this study, we have proposed our method to detect IoT attacks that depend on ML-based approaches. Also, we implemented

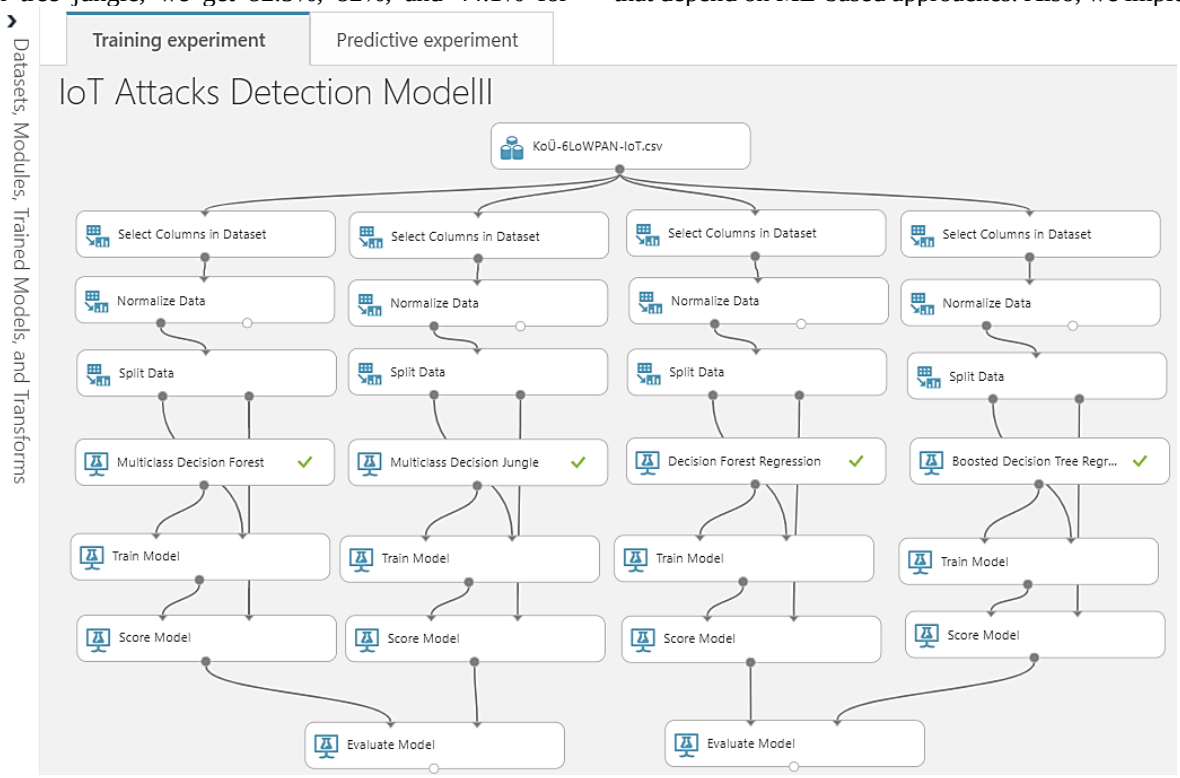

Figure 10. Decision tree-based Models

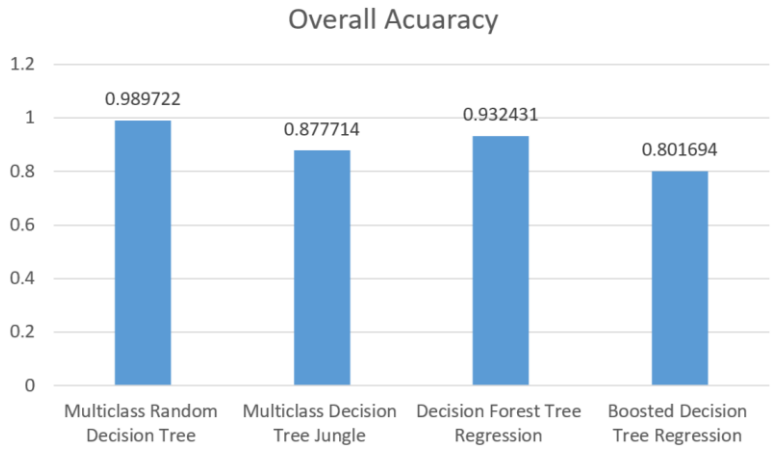

Figure 11. The overall accuracy of decision tree-based model.

and examined three attacks in the IoT ecosystem: DoS, BHA, and OOA, to generate a novel dataset of IoT features-based that was produced via Cooja simulator-based. The ML-based approaches depend on decision tree-based models that have proven their efficiency in manipulating, examining, and classifying malicious activity of the IoT features generated via the Cooja simulator. As a result, the multiclass Random Decision Forest ML-based model achieved 98.9% overall accuracy in detecting IoT attacks for the KoÜ-6LoWPAN-IoT dataset compared to the decision tree jungle, decision forest tree regression, and boosted decision tree regression, which achieved 87.7%, 93.2%, and 87.1%, respectively. The multiclass random decision forest achieved the highest accuracy overall. As the orientation of designing the lightweight ML custom algorithm to solve the IoT security problem. It is a matter of interest and tends to be on the minds of developers.

References

[1] K. Wu, R. A. Laghari, M. Ali, and A. Ayub Khan. "A Review and State of Art of Internet of Things (IoT)." Archives of Computational Methods in Engineering, pp. 1-19,2021.

[2] I. Lee, and K. Lee. "The Internet of Things (IoT): Applications, investments, and challenges for enterprises." Business horizons, pp. 431-440, 2015.

[3] C. Bayılmış, M. A. Ebleme, Ü. Çavuşoğlu, K. Küçük, A. Sevin, A survey on communication protocols and performance evaluations for Internet of Things, Digital Communications and Networks, 2022, https://doi.org/10.1016/j.dcan.2022.03.013.

[4] T. Pflanzner, A. Kertesz. "A taxonomy and survey of IoT cloud applications." EAI Endorsed Transactions on Internet of Things: Terjedelem-14. 2018.

[5] C. Bayilmis, K. Kucuk, Internet of Things: Theory and Applications, Daisy Science Publishing (2019).

[6] M. S. Mekala and P. Viswanathan, "A Survey: Smart agriculture IoT with cloud computing," 2017 International conference on Microelectronic Devices, Circuits and Systems (ICMDCS), pp. 1-7, 2017.

[7] L. Minh Dang, M. Jalil Piran, D. Han, K. Min and H. Moon "A survey on internet of things and cloud computing for healthcare." Electronics, 2019.

[8] A. A. Laghari, H. He, A. Khan, N. Kumar and R. Kharel, "Quality of Experience Framework for Cloud Computing (QoC)," in IEEE Access, vol. 6, pp. 64876-64890, 2018.

[9] R. Sikarwar, P. Yadav and A. Dubey, "A Survey on IOT enabled cloud platforms," 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), pp. 120-124, 2020.

[10] Ping, H., Wang, J., Ma, Z., & Du, Y. "Mini-review of application of IoT technology in monitoring agricultural products quality and safety". Int. Journal of Agricultural and Biological Engineering, 11(5), pp. 35-45. 2018.

[11] F. T. Johnsen et al., "Application of IoT in military operations in a smart city," 2018 International Conference on Military Communications and Information Systems (ICMCIS), 2018, pp. 1-8.

[12] K. Riad, T. Huang, L. Ke."A dynamic and hierarchical access control for IoT in multi-authority cloud storage." Journal of Network and Computer Applications. pp. 102633. 2020.

[13] N. M. Abdulkareem,S. R. M. Zeebaree,M. A. M.Sadeeq, D. M. Ahmed,A. S. Sami,R. R. Zebari,"IoT and Cloud computing issues, challenges and opportunities: A review." Qubahan Academic Journal, pp.1-7. 2021.

[14] A. Ramakrishnan, D. Preuveneersa, Y. Berbersa,"Enabling self-learning in dynamic and open IoT environments." Procedia Computer Science, pp. 207-214. 2014.

[15] D. Choudhary "Security Challenges and Countermeasures for the Heterogeneity of IoT Applications." Journal of Autonomous Intelligence, pp. 16-22. 2019.

[16] Y. Hajjaji, A. Boulilaac, I. R. Faraha, I. Romdhanib, A.Hussain, "Big data and IoT-based applications in smart environments: A systematic review." Computer Science Review, pp. 100318. 2021.

[17] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications." IEEE communications surveys & tutorials, pp. 2347-2376,2015.

[18] A. Verma and V. Ranga, "Security of RPL Based 6LoWPAN Networks in the Internet of Things: A Review," in IEEE Sensors Journal, vol. 20, no. 11, pp. 5666-5690, 1 June1, 2020.

[19] A. Rghioui, M. Bouhorma and A. Benslimane, "Analytical study of security aspects in 6LoWPAN networks," 2013 5th

International Conference on Information and Communication Technology for the Muslim World (ICT4M), 2013, pp. 1-5.
[20] L. Xiao, X. Wan, X. Lu, Y. Zhang and D. Wu, "IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security?," in IEEE Signal Processing Magazine, vol. 35, no. 5, pp. 41-49, Sept. 2018.
[21] M. Injadat, A. Moubayed and A. Shami, "Detecting Botnet Attacks in IoT Environments: An Optimized Machine Learning Approach," 2020 32nd International Conference on Microelectronics (ICM), 2020, pp. 1-4.
[22] R. Barker, "Security aspects in 6lowPan networks," 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010), 2010, pp. 660-660.
[23] M. Seliem and K. Elgazzar, "IoTeWay: A Secure Framework Architecture for 6LoWPAN Based IoT Applications," Conf. on IEEE, Egypt, Jan. 2019.
[24] E. Kim, D. Kaspar, JP. Vasseur, "Design and application spaces for IPv6 over low-power wireless personal area networks (6LoWPANs)." RFC6568 (2012).
[25] Osorio, O. Gracia, B. S. Reyes Daza, and O. J. Salcedo Parra. "Comparative Study of Performance for 804.15. 4 ZigBee and 6LoWPAN Protocols." SOFSEM (Student Research Forum Papers/Posters). Vol. 1548. 2016.
[26] Tanveer, Muhammad, et al. "S6AE: Securing 6LoWPAN using authenticated encryption scheme." Sensors, 2020.
[27] P. P. Ioulianou and V. G. Vassilakis, "Denial-of-service attacks and countermeasures in the RPL-based Internet of Things." Computer Security. Springer, Cham,374-390, 2019.
[28] R. Dobbins, D. Migault, R. Moskowitz, N. Teague, L. Xia, and K. Nishizuka, "Use Cases for DDoS Open Threat Signalling," RFC 8903, 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8903.
[29] N. Abughazaleh, R. Bin, M. Btish, and H. M., "DoS Attacks in IoT Systems and Proposed Solutions," IJCA, vol. 176, no. 33, pp. 16–19, Jun. 2020.
[30] V. Neerugatti and A. R. M. Reddy, "Detection and prevention of black hole attack in RPL protocol based on the threshold value of nodes in the internet of things networks." Int. J. Innov. Technol. Explor. Eng 8.9 ,2018.
[31] S. Ali, M. A. Khan, J. Ahmad, A. W. Malik, and A. U. Rehman, "Detection and prevention of Black Hole Attacks in IoT & WSN," 3rd Inter. Conf. on FMEC, Spain, pp. 217–226, May 2018.
[32] F. Ahmed and Y. B. Ko, "Mitigation of black hole attacks in Routing Protocol for Low Power and Lossy Networks," Security and Commun. Net., vol. 9, no. 18, pp. 5143–5154, Dec. 2016.
[33] A. H. Farea and K. Küçük, "Enhancement Trust Management in IoT to Detect ON-OFF Attacks with Cooja", International Journal of Multidisciplinary Studies and Innovative Technologies, vol. 5, no. 2, pp. 123-128, Nov. 2021.
[34] C. V. L. Mendoza and J. H. Kleinschmidt, "Mitigating on-off attacks in the internet of things using a distributed trust management scheme," IJDSN, vol. 2015, 2015.
[35] A. A. Wardana, "IoT Object Security towards On-off Attack Using Trustworthiness Management," 8th inter. Conf. on ICoICT, Indonesia, 2020.
[36] M. Hasan, M. Milon Islam, M. Zarif, and M. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," IoT, vol. 7, Sep. 2019.
[37] P.Bedia, S. Mewadab, R. Arjunarao, V. Chaitanya, S. Kanwalvir, S. Dhindsae, M. Ponnusamyf, R. Sikarwarg, "Detection of attacks in IoT sensors networks using machine learning algorithm," Microprocessors and Microsystems, vol. 82, Apr. 2021.
[38] M. Waqas, K. Kumar, A. Laghari,U. Saeed,M. Rind. "Botnet attack detection in Internet of Things devices over cloud environment via machine learning." Concurrency and Computation: Practice and Experience 34, no. 4, 2022.
[39] https://www.kaggle.com/datasets/francoisxa/ds2ostraffictraces
[40] https://www.unb.ca/cic/datasets/index.html
[41] https://research.unsw.edu.au/projects/unsw-nb15-dataset
[42] https://research.unsw.edu.au/projects/bot-iot-dataset
[43] I. A. Ismaili, A. Azyat, N.Raissouni, N. B. Achhab, A. Chahboun M. Lahraoua "Comparative study of ZigBee and 6LoWPAN protocols." ICCWCS 2019: Third International Conference on Computing and Wireless Communication Systems, April 24-25 , 2019.
[44] M.V.R Jyothisree and S.Sreekanth "Attacks in RPL and Detection Technique used for Internet of Things." Inter. Jou. Recent Techno. and Eng., vol. 8, 2019, ISSN: 2277-3878.
[45] "Contiki 6LoWPAN Quick Guide Contiki on STM32 Nucleo plugged with Sub-1 GHz RF expansion board (X-NUCLEO-IDS01A4, X-NUCLEO-IDS01A5).", Apr. 2016. [Online]. Available: https://github.com/contiki-os/Contiki
[46] C. Hennebert and J. dos Santos, "Security Protocols and Privacy Issues into 6LoW-PAN Stack: A Synthesis," IEEE Internet of things journal, vol. 1, no. 5, pp. 384–398, 2014.
[47] Y. Chen, J. Chanet, K. Hou, Y. Chen, J.-P. Chanet, and K. Mean HOU, "RPL Routing Protocol a case study: Precision agriculture," CF-WoFUCT 2012, Feb. 2012. [Online] Available: https://hal.archives-ouvertes.fr/hal-00681319
[48] A. S. J. Charles and P. Kalavathi, "QoS Measurement of RPL using Cooja Simulator and Wireshark Network Analyser View project QoS Measurement of RPL using Cooja Simulator and Wireshark Network Analyser," IJCSE, vol. 6, 2018, ISSN: 2347-2693.
[49] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes, "Powertrace: Network-level Power Profiling for Low-power Wireless Networks," SICS Technical Report, T2011, Mar. 2011, ISSN 1100-3154.