

Article

Implementation of Lightweight Machine Learning-Based Intrusion Detection System on IoT Devices of Smart Homes

Abbas Javed ¹, Amna Ehtsham ¹, Muhammad Jawad ^{1,2}, Muhammad Naeem Awais ¹, Ayyaz-ul-Haq Qureshi ^{3,*} and Hadi Larijani ⁴

¹ Department of Electrical and Computer Engineering, COMSATS University Islamabad, Lahore Campus, Punjab 54000, Pakistan; abbasjaved@cuilahore.edu.pk (A.J.); amna.ehtsham@gmail.com (A.E.); mjawad@cuilahore.edu.pk (M.J.); naeem.awais@cuilahore.edu.pk (M.N.A.)

² Hitachi Energy Research, Pawia 7, 31-154 Kraków, Poland

³ Department of Cyber Security and Networks, School of Computing, Engineering and Built Environment, Glasgow Caledonian University, Glasgow G4 0BA, UK

⁴ SMART Technology Research Centre, Department of Cyber Security and Networks, School of Computing, Engineering and Built Environment, Glasgow Caledonian University, Glasgow G4 0BA, UK; h.larijani@gcu.ac.uk

* Correspondence: aqu2@gcu.ac.uk

Abstract: Smart home devices, also known as IoT devices, provide significant convenience; however, they also present opportunities for attackers to jeopardize homeowners' security and privacy. Securing these IoT devices is a formidable challenge because of their limited computational resources. Machine learning-based intrusion detection systems (IDSs) have been implemented on the edge and the cloud; however, IDSs have not been embedded in IoT devices. To address this, we propose a novel machine learning-based two-layered IDS for smart home IoT devices, enhancing accuracy and computational efficiency. The first layer of the proposed IDS is deployed on a microcontroller-based smart thermostat, which uploads the data to a website hosted on a cloud server. The second layer of the IDS is deployed on the cloud side for classification of attacks. The proposed IDS can detect the threats with an accuracy of 99.50% at cloud level (multiclassification). For real-time testing, we implemented the Raspberry Pi 4-based adversary to generate a dataset for man-in-the-middle (MITM) and denial of service (DoS) attacks on smart thermostats. The results show that the XGBoost-based IDS detects MITM and DoS attacks in 3.51 ms on a smart thermostat with an accuracy of 97.59%.

Keywords: intrusion detection system; embedded machine learning; TinyML; internet of things; machine learning; cloud computing; edge machine learning



Citation: Javed, A.; Ehtsham, A.; Jawad, M.; Awais, M.N.; Qureshi, A.-u.-H.; Larijani, H. Implementation of Lightweight Machine Learning-Based Intrusion Detection System on IoT Devices of Smart Homes. *Future Internet* **2024**, *16*, 200. <https://doi.org/10.3390/fi16060200>

Academic Editors: Olivier Markowitch and Jean-Michel Dricot

Received: 1 May 2024

Revised: 30 May 2024

Accepted: 1 June 2024

Published: 5 June 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Despite the impact of COVID-19, the number of Internet of Things (IoT) devices grew by 9% as observed in a 2021 report [1]. It is estimated that the number of IoT devices will surpass 27 billion by 2025 [1]. The reason behind this boom can be attributed to the easy availability of internet connectivity and low-cost sensors. While these devices have made our lives easier, they have also created a window of opportunity for hackers to exploit privacy. Some hacked systems, such as medical devices, can even be life-threatening. Smart home devices, when hacked, can also compromise the privacy of users. Every layer of the IoT, including the application, network, middleware, and sensing layer, is susceptible to attacks [2]. Intrusion detection systems (IDSs) designed for IoT systems aim to detect and respond to these threats in real time, ensuring the security and integrity of the IoT ecosystem. Nowadays, the use of IoT devices in our homes is increasingly geared towards creating a more comfortable living environment. These smart home devices communicate with one another using Zigbee, Bluetooth, and WiFi, and connect to the cloud via internet access (typically through a WiFi router) [3]. They are usually controlled remotely through

cloud services and mobile apps. However, this remote accessibility of IoT devices in smart homes presents numerous opportunities for attackers, especially considering that most occupants have limited knowledge of security measures [4]. Securing such devices proves challenging and complex due to their limited storage and computational power.

Several machine learning and deep learning algorithms have been studied for intrusion detection [5–9]. All the aforementioned papers discuss a centralized approach. A few researchers also presented distributed approaches for intrusion detection in IoT applications [10–13].

Researchers utilized fog computing in [13–16] for implementing IDSs in fog but employed desktop PCs for the implementation. Similarly, in [17], the authors employed Raspberry Pi for binary class IDS implementation on a fog node. Raspberry Pi-based edge devices were also used for IDS implementation in [18,19]. In [20], the authors utilized Google Edge TPU and Raspberry Pi for IDS implementation on the edge, while in [21], the authors developed an IDS for mobile edge computing (MEC), but the proposed technique was tested on a desktop computer. Due to limited resources, the authors implemented a binary class IDS on an edge device [20,21].

The researchers used edge devices (Raspberry Pi/Desktop PCs/gateways), fog, and the cloud for implementing the IDS due to the availability of network traffic analyzers, such as Wireshark (<https://www.wireshark.org> (accessed on 26 April 2024)) and tcpdump (<https://www.tcpdump.org/> (accessed on 26 April 2024)) but an ML-based IDS is not embedded in IoT devices. Due to limited processing power, the lightweight IP (lwIP) (<https://github.com/espressif/esp-lwip> (accessed on 26 April 2024)) library for ESP32 (<https://www.espressif.com/en/products/socs/esp32> (accessed on 26 April 2024)) has restricted access to network features. Consequently, implementing an IDS on a microcontroller-based device without using network traffic analyzer software is an exceedingly challenging task. In this paper, we have implemented a real-world scenario in which we employed a smart thermostat using ESP32, an HTTP page on a cloud server (<http://16.170.215.67/display> (accessed on 26 April 2024)), and a Raspberry Pi-based adversary node for generating denial of service (DoS) and man-in-the-middle (MITM) attacks to collect a dataset called the intrusion detection in smart homes (IDSH) dataset [22]. The dataset is collected by using lwip and socket library for an ESP32 microcontroller. An ML-based binary class IDS is embedded in a smart thermostat using TinyML [23], which is responsible for monitoring room temperature and controlling HVAC systems. Additionally, this smart thermostat connects to WiFi and uploads air temperature and humidity data to a hosted web page on a cloud-based web services server, where an ML-based multiclass IDS is also deployed. To enhance the security of the microcontroller-based smart thermostat, we propose deploying a lightweight IDS based on XGBoost. Considering that IoT devices in smart homes connect to WiFi routers, they become susceptible to various security risks, including DoS, DDoS, MITM, ransomware, and scanning attacks. The main research objective of this research article is to develop an IDS for IoT devices (a smart thermostat in our case) of smart homes which are directly communicating with web servers through WiFi access points and additional hardware (Raspberry Pi-based edge device) is not available. Therefore, in this work, we proposed a two-layered ML-based IDS which can protect the smart thermostat by embedding an IDS in the smart thermostat and can also protect the cloud by implementing the IDS on the cloud.

The main contributions of this paper are as follows:

- A novel bi-directional computationally efficient ML-based decentralized multiclass IDS is proposed for the deployment on IoT devices and the cloud.
- An embedded XGBoost-based IDS for real-time binary classification of attacks in smart thermostats using TinyML is proposed. Moreover, a thorough comparison is made for ANN-, XGBoost-, RF-, and DT-based IDS implementation on low-cost microcontroller-based IoT devices for binary classification. The algorithms are compared in terms of accuracy, inference time, and storage requirement.

- A computationally efficient XGBoost IDS is implemented on the cloud for monitoring the incoming traffic from IoT devices and external attacks. The XGBoost IDS is implemented for the identification of DoS, DDoS, MITM, port scanning, injection, backdoor, XSS, and ransomware attacks. Moreover, a detailed comparison of different ML algorithms is made for IDS implementation on the cloud side in terms of accuracy, F1-score, precision, recall, inference time, and computational complexity.
- To conduct real-time testing and validation of the IDS deployed on a smart thermostat, we developed a Raspberry Pi-based adversary specifically designed to generate DoS and MITM attacks on the smart thermostat. We collected a new dataset called the intrusion detection in smart homes (IDSH) dataset [22] comprising 4144 samples, encompassing normal activities, DoS attacks, and MITM attacks on the smart thermostat.

The rest of the paper is organized as follows: Section 2 presents the literature review on centralized and decentralized IDS. In Section 3, the proposed methodology is explained. Section 4 discusses the results along with a comparative analysis. The paper is concluded in Section 5 along with future research directions.

2. Related Work

An IoT system is vulnerable to different types of attacks [2]. These attacks are explained in detail in [24] and include password cracking attacks, ransomware, port scanning, MITM attacks, DoS/DDoS attacks, backdoors, injection attacks, and XSS attacks. A password cracking attack is launched against a device to guess its password using a commonly available set of passwords online. A ransomware attack is launched to lock a device's data and demand money in exchange for the data. A port scanning attack is used to check which port on a device is open to launch an attack against that port. An MITM attack is used to intercept traffic between any two nodes in the network. A DoS/DDoS attack is used to occupy bandwidth in a network. For example, a device or a set of devices is directed toward a single server to send false queries, making the server unresponsive to the actual queries. A backdoor attack is launched to use/hack devices without anyone knowing. A hacker who launches a backdoor attack has full access to the device. Injection/XSS attacks are launched against web pages to alter their data.

Recently, in [25], the authors calculated the energy consumption of running DT-, RF-, and ANN-based ML models for intrusion detection in IoT devices using the TinyML library. However, the energy consumption is based on theoretical calculations, and the ML models were not tested for real-world intrusion detection systems. The authors in [26] developed an IoT device dataset by generating a botnet attack on a smart health service testbed. The dataset is labeled as normal and attack traffic. The authors utilized Wireshark for collecting the dataset. The results show that the accuracy of an RF-based IDS on the collected dataset is 99.98%.

In [9], the TON_IoT dataset [27] is used to train multiple ML algorithms for intrusion detection in intelligent transportation systems, where the XGBoost algorithm outperforms other ML algorithms with an accuracy of 99.92%. In [28], the authors developed an enhanced anomaly-based intrusion detection method (EIDM) for classifying 15 attacks with an accuracy of 95%. The first application of random neural networks (RNNs) for IDSs was proposed in our previous work [29] in which an RNN with an artificial bee colony algorithm is used for binary classification of intrusion detection in the cloud. In [30], the authors developed an IDS using the hybrid deep learning model. The proposed technique achieved 96% accuracy, but it incurred a high computational burden. In [31], a five-stack ResNet was used as a deep learning method, trained on two different datasets: N-BaIoT and the power system dataset.

The authors in [32] proposed to implement ensemble ML model selection on the cloud and IDS implementation on fog for real-time prediction to reduce the attack detection time. In [33], a distributed IDS approach is presented. Attacks were classified as normal or attack at the fog node. If identified as an attack, further classification occurs at the cloud

node. The DNN-KNN algorithm showcased accuracies of 99.77% and 99.85% on NSL-KDD and CICIDS2017, respectively. In [34], a fog computing-based distributed IDS approach is introduced. Data are analyzed for attacks at the fog node, and if detected, relevant information is stored in the cloud server for subsequent action. In [16], an anomaly-based IDS is moved to fog nodes, and the Distributed Smart Space Orchestration System (DS2OS) dataset is used to train different ML algorithms. RF showed an accuracy of 99.99%. A distributed IDS using fog computing is designed in [35] to detect DDoS attacks in the memory pool of a blockchain-enabled IoT network. The performance of the proposed IDS is evaluated using machine learning algorithms (RF and XGBoost) and evaluation metrics on an actual IoT-based BoT-IoT dataset containing various recent botnet attacks.

In [19], the authors presented Passban, an intelligent IDS for edge devices. The IDS was deployed on an IoT gateway developed with Raspberry Pi 3. Passban successfully detected HTTP and SSH brute force, port scanning, and SYN flood attacks. Another IDS, named IoT-KEEPER, is proposed in [18], capable of detecting MITM, DoS, and scanning attacks at the IoT gateway. The authors tested IoT-KEEPER's performance on Raspberry Pi 3.

In [36], an IDS is introduced to secure edge IoT devices, operating in two phases. In the first phase, features are extracted and then these features are trained using the DNN algorithm. This system presents an accuracy of 99.23% and an F1-score of 99.27%. In [37], an edge-of-things computing-based intrusion detection mechanism is presented. The best accuracy achieved by this network is 88.4%.

The literature review above has discussed both centralized and decentralized approaches deployed on the edge, fog, and cloud sides. To effectively implement an IDS on the edge and fog sides, it needs to be both accurate and computationally efficient in promptly detecting attacks. Successfully detecting intrusions on the edge/fog side of an IoT network could proactively save cloud resources from further proliferation. Additionally, detailed studies on IDSs for IoT devices, developed with microcontrollers having limited computation resources, are lacking. There has been no study evaluating the feasibility of embedding an IDS on both IoT devices and the cloud side.

3. Materials and Methods

This research investigated a multilayered approach to classify intrusions that an IoT network might face. The goal is to distribute the load between the cloud and edge sides of the IoT network for intrusion classification. After load balancing, the TON_IoT dataset [27] was divided according to attacks on the edge and cloud sides of the IoT network. Multiple ML algorithms were then trained on the dataset to select the one that shows the best results. In this work, we considered an application of a smart thermostat that uploads temperature, humidity levels, and air conditioner status to a webpage hosted on the Amazon Web Service (AWS) cloud server. The proposed technique involves a two-layered IDS, which begins intrusion detection at the device level. The first layer of the IDS is deployed on a low-cost ESP32 microcontroller-based smart thermostat controller for binary classification. This IDS on the smart thermostat detects DoS, MITM, ransomware, and scanning attacks.

The second layer of the IDS is deployed on the cloud side to classify DoS, DDoS, brute force, injection, XSS, ransomware, port scanning, MITM, and backdoor attacks. The architecture of the proposed distributed approach is shown in Figure 1. The approach is evaluated in real time by implementing the adversary node using Raspberry Pi 4 (manufactured by the Raspberry Pi Foundation, Cambridge, United Kingdom.) with Kali Linux OS (version: 2023.3) The IDS implementation on the smart thermostat is evaluated in real time by generating attacks from the adversary node targeted at the smart thermostat.

For real-time testing of the IDS on the smart thermostat, DoS and MITM attacks are generated to disrupt the communication between the smart thermostat and the AWS cloud server.

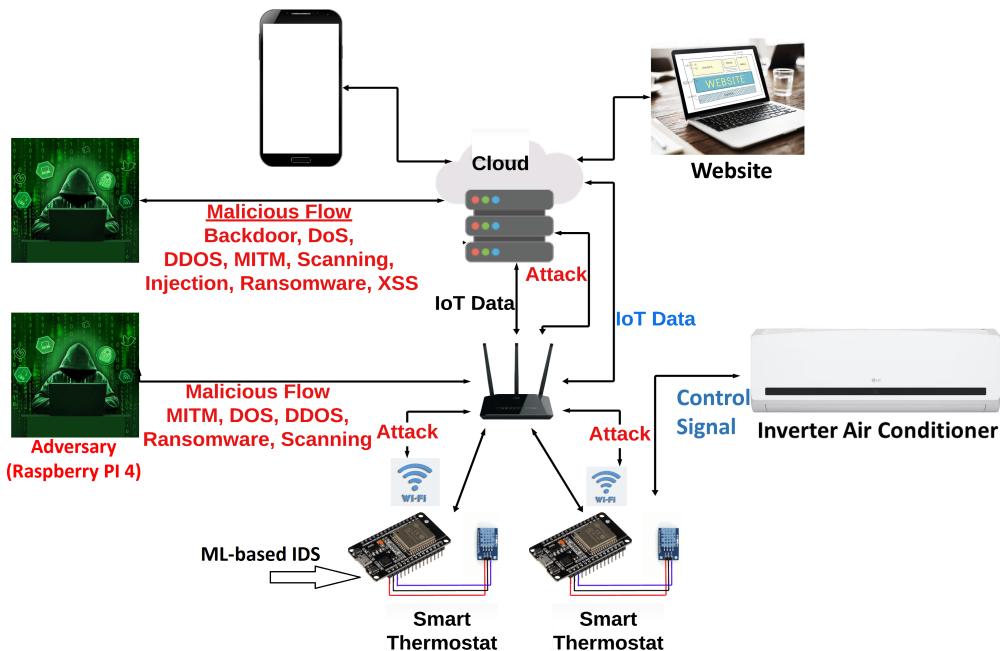


Figure 1. System architecture for distributed IDS.

3.1. Threat Model

In this work, an adversary seeks to disrupt, intercept, and manipulate the communication between the smart thermostat and the cloud. The adversary considered in this work has no knowledge of the ML-based IDS deployed on the smart thermostat.

3.1.1. Targeted Model

The smart thermostat monitors the temperature and humidity of the building and uploads the indoor environment data to the cloud. The user can send the control signal of the inverter air conditioner from the website to the smart thermostat. The bi-directional communication between the smart thermostat and the cloud is established through WiFi.

3.1.2. Adversarial Capabilities

The adversary possesses the capability to enable various attack as shown in Figure 1. Being connected to the same hotspot as the smart thermostat allows the adversary to intercept and disrupt communication between the thermostat and the cloud. The adversary is proficient in launching the following attacks: DoS, MITM, scanning, and ransomware.

3.1.3. Adversary Goals

The primary objective of the adversary is to intercept the communication between the smart thermostat and the cloud. The adversary goals are as follows:

- Executing DoS attacks to overwhelm the smart thermostat or cloud server, rendering them unresponsive.
- Intercepting and tampering with the data transmission between the smart thermostat and the cloud through MITM attacks.
- Attempting to exploit vulnerabilities in the smart thermostat or cloud infrastructure via scanning or injection attacks.
- Potentially deploying ransomware to encrypt data or compromise the operation of the smart thermostat or cloud server.

3.2. Training Dataset for IDS

3.2.1. TON_IoT Dataset

This is a heterogeneous dataset specifically captured for intrusion detection in IoT applications totaling 461,043 instances. The edge layer of the testbed comprises sensors, weather stations, smart TVs, gateways, and Virtual Machines (VMs) managing the network. The fog layer includes client and vulnerable VMs, along with offensive Kali systems. Additionally, the fog layer hosts a middleware server that manages seven IoT sensors and their message queuing telemetry transport (MQTT) services using the Node-RED tool. The cloud layer encompasses servers simulating communication with the cloud, a vulnerable website for attack simulations, and an MQTT broker. This testbed covers various attack categories, including port scanning, DoS/DDoS, ransomware, backdoor, data injection, cross-site scripting, brute force, and man-in-the-middle attacks. A total of 45 features are extracted from this testbed.

The main points considered for selection of the dataset are as follows:

- Testbed architecture;
- Number of features;
- Number of attacks simulated.

The MQTT IoT dataset simulates only data of sensors that are communicating over MQTT and there are only four attack categories covered [38]. The Bot-IoT dataset focuses more on the network and there are a fewer number of attacks covered [39]. In the TON_IoT dataset, the testbed is heterogeneous and mimics the behavior of the edge, fog, and cloud side very well. Moreover, this testbed includes seven IoT sensors, and it covers a wide range of attacks that proved helpful while doing load balancing for the distributed approach for the IDS.

3.2.2. Load Balancing

The cloud has enough computational resources, and any complex deep learning algorithm can be trained and deployed on the cloud; therefore, it is decided to deploy a classifier that classifies DoS, DDoS, brute force, injection/XSS attacks, ransomware, port scanning, and backdoor attacks.

On the other hand, DoS/DDoS, MITM, ransomware, scanning, and backdoor can affect the smart thermostat, so a classifier deployed in the smart thermostat will protect it from potential attacks. For this reason, the datasets for MITM, DoS, and Normal traffic are converted into binary class problems.

3.2.3. Data Preprocessing

The TON_IoT dataset is selected to be used in our research. However, the dataset needed preprocessing to be further used for classification. We employed feature scaling to normalize all input features. We used the standard minmax function for data normalization that converts all data points of the column in the range of [0,1] or [-1,1] depending on all positive data entries or those having negative values. Moreover, the output variable include string parameters that must be converted into an integer array to be further used in the ML algorithms. Therefore, label encoding is used for the output.

3.2.4. Feature Selection and Feature Extraction

In this work, we utilized chi-square [40] for feature selection (FS) and Principal Component Analysis (PCA) for feature extraction (FE) [41] in order to reduce the features. The features selected by chi-square are shown in Table 1. PCA transforms original features into new, uncorrelated features (principal components) that capture the variance in data. The first few principal components usually capture most of the variance, allowing for effective dimensionality reduction while preserving the data's essential structure. In this work, we tested PCA for FE using 10, 15, and 25 components.

Table 1. Selected features.

No. of Features	Selected Features
10	src_port, dst_port, src_bytes, dst_bytes, missed_bytes, src_ip_bytes, dst_ip_bytes, dns_query, dns_qclass, http_response_body_len,
15	src_port, dst_port, service, duration, src_bytes, dst_bytes, missed_bytes, src_pkts, src_ip_bytes, dst_pkts, dst_ip_bytes, dns_query, dns_qclass, dns_qtype, http_response_body_len
25	src_port, dst_port, proto, service, duration, src_bytes, dst_bytes, conn_state, missed_bytes, src_pkts, src_ip_bytes, dst_pkts, dst_ip_bytes, dns_query, dns_qclass, dns_qtype, dns_rcode, dns_AA, dns_RD, dns_RA, dns_rejected, http_request_body_len, http_response_body_len, http_status_code, http_user_agent,

3.3. Implementation of Machine Learning-Based Intrusion Detection System on Cloud

Different ML algorithms are implemented to check their performance on the cloud side of an IoT network. First, the data are preprocessed and divided into training, testing, and validation datasets. Then, five different ML algorithms are trained on the data. After training, the algorithms are tested on the test dataset. In this work, we implemented the XGBoost-, LSTM-, Conv1d-, RF-, and MLP-based IDSs for deployment on the cloud.

3.4. Implementation of Adversary for Real-Time Testing and IDSH Dataset Collection

The ToN_IoT dataset contains thermostat data, including temperature, thermostat status, timestamp, date, label, and type. However, this dataset is not generalized and may not be suitable for smart home applications operating in different environmental conditions. In addition to sensor data, every data packet sent through a network by an IoT device contains additional information, such as connectivity and network protocols. This provides raw information about the internal and external activity of these IoT devices, complementing sensor data. In this study, we used network features, such as connection activity, statistical activity, and DNS activity, to train the IDS deployed on IoT devices.

For IDS implementation on the IoT devices in our smart home, specifically, the smart thermostat, we generated our own dataset named the IDSH dataset, available on Kaggle [22]. This dataset combines sensor data, thermostat status, and network features (i.e., connection activity and statistical activity) for MITM, DoS, and normal traffic.

The smart thermostat is designed to monitor the indoor temperature and humidity of the room and control the air conditioner by sending ON/OFF signals. The temperature and humidity are measured by the thermostat using a DHT22 sensor. These measurements are sent to a web application via an HTTP POST request, which receives the data and displays the indoor temperature on the webpage. Additionally, the user can control the air conditioner remotely through the web application by sending ON/OFF commands.

In this work for testing the accuracy of an IDS deployed on a smart thermostat, the adversary is implemented with Kali Linux scripts running on Raspberry Pi 4. The attacker node generates the MITM, and DoS attacks on the smart thermostat communicating with the cloud server for data uploading and control signal input. The attacker node carried out MITM using Address Resolution Protocol (ARP) spoofing and DoS attacks on the smart thermostat by flooding them with the high volume of traffic and requests. The smart thermostat does not have access to packet analyzer tools like Wireshark and tcpdump for extracting network features. Therefore, we utilized the Esp32 lwIP library and socket library to extract real-time network parameters. The block diagram of the smart thermostat for collecting the dataset is shown in Figure 2. The parameters of the dataset collected at each step are shown in Figure 2.

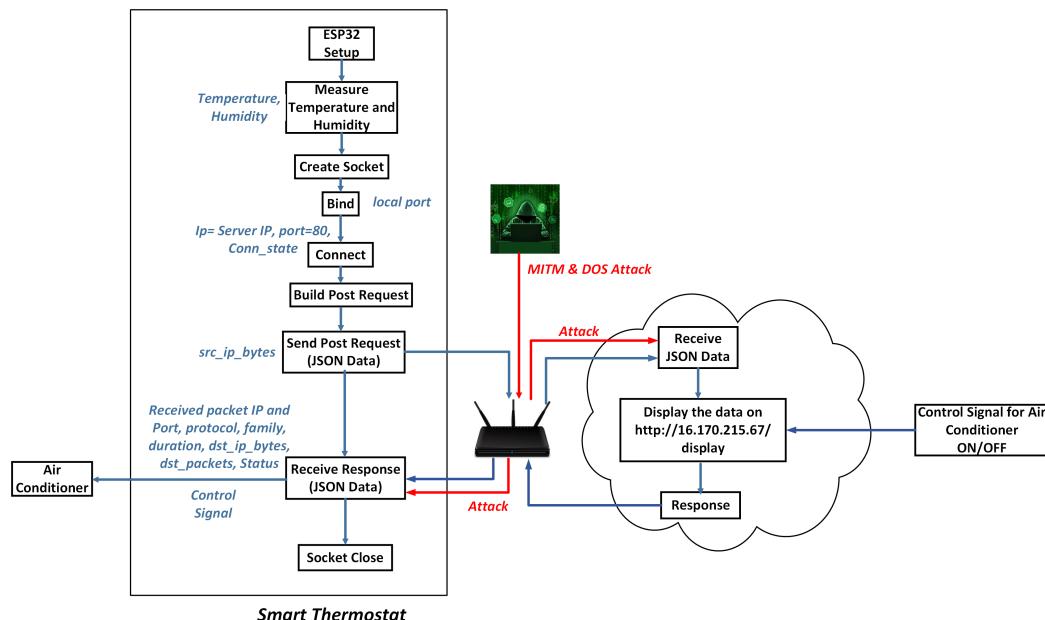


Figure 2. Dataset collection on smart thermostat.

The steps involved in dataset collection are as follows:

1. Setup the microcontroller by including header files for the lightweight IP and socket (`lwip/socket.h`).
2. Create, bind, and connect the server using the server IP and port.
3. Build a POST request and send the POST request to the web server.
4. The web server receives the POST request, displays the data, and responds to the smart thermostat with the status of the air conditioner (i.e., ON/OFF).
5. The smart thermostat receives the response and collects the features of the collected dataset as shown in Table 2.

For MITM and DoS attacks, the adversary generates an MITM attack using the `arpspoof` command and a DoS attack using `hping3`. During the attack, we collected the dataset from the smart thermostat by following steps 1–5. We collected an IDSH dataset comprising 4144 samples, encompassing instances of normal traffic, DoS attacks, and MITM attacks. Among these, 2762 samples correspond to normal traffic, 292 samples are of DoS attacks, and 1090 samples represent MITM attacks. In this work, we used ID 21 (Label) as the target for binary classification, while for multiclass classification, ID 22 (Type) could be used as the target. We used IDs 3 and 5–20 as input features, excluding the timestamp (ID 1), source IP (ID 2), and destination IP (ID 4) from the input features to avoid overfitting.

Table 2. Description of IDSH dataset.

ID	Feature	Type	Description
1	timestamp	Time	Timestamp of connection
2	source_ip	String	Source IP address
3	source_port	Number	Source ports which originate endpoint's TCP/UDP ports
4	destination_ip	String	Destination IP address
5	dst_port	Number	Destination ports which respond to endpoint's TCP/UDP ports
6	protocol	String	Transport layer protocols of flow connections

Table 2. *Cont.*

ID	Feature	Type	Description
7	Family	String	Address family
8	Service	String	Dynamically detected protocols, such as DNS, HTTP, and SSL
9	Duration	Time	The difference in the time between the packet being sent from the source and receiving the response from the cloud server
10	source_bytes	Number	Source bytes which originate from the source
11	destination_bytes	Number	Destination bytes which are responses from the destination
12	conn_state	String	Various connection states, such as S0 (connection without replay), S1 (connection established), and REJ (connection attempt rejected)
13	missed_bytes	Number	Number of missing bytes in content gaps
14	source_packets	Number	Number of original packets which is estimated from source systems
15	src_ip_bytes	Number	Number of original IP bytes which is the total length of IP header field of source systems
16	destination_packets	Number	Number of destination packets which is estimated from destination systems
17	dst_ip_bytes	Number	Number of destination IP bytes which is the total length of IP header field of destination systems
18	Temperature	String	Room temperature ($^{\circ}\text{C}$) of the Zone
19	Humidity	String	Humidity (%) of the Zone.
20	Status	String	Status of air conditioner (ON/OFF).
21	Label	Number	Tag Normal traffic as 0 and attack as 1.
22	Type	String	Tag attack categories as Normal, DoS, and MITM.

3.5. Implementation of Machine Learning-Based Intrusion Detection System on Smart Thermostat (IoT Device)

As a feasibility study for the implementation of an IDS on a smart thermostat, two models of the IDS are trained using the TON_IoT dataset. The IDS is first trained with connection activity, statistical activity, and DNS activity features. The second model is trained with connection activity and statistical activity. To avoid overfitting, the source and destination IP addresses and timestamps are removed from the training and testing dataset.

Four different ML algorithms, XGBoost, DT, RF, and ANN, are deployed on the microcontroller (ESP32)-based smart thermostat for binary classification of attacks. The ML-based IDS is deployed on a microcontroller using the TinyML [23] library.

4. Results

This section covers a discussion of various ML algorithms used for implementing an IDS on the cloud, followed by the presentation of ML-based IDS results on IoT devices (smart thermostat). Additionally, this section provides details on the implementation of ML algorithms on a microcontroller using the TinyML library [23]. Various ML models are trained to be deployed on the cloud side of an IoT network with and without FS techniques. The TON_IoT dataset is used to train the XGBoost, LSTM, Conv1d, MLP, and RF algorithms and their results are discussed in this section. For implementation on a smart thermostat (edge device), XGBoost is compared with DT, RF, and ANN in terms of accuracy, storage requirement, and computational delay. The highlighted values, in bold in the tables, denote the models with superior performance. Finally, the accuracy of the IDS is also evaluated for MITM and DoS attacks on the smart thermostat generated by the adversary.

4.1. Machine Learning-Based IDS on the Cloud without FS Method

The LSTM, Conv1d, XGBoost, RF, and MLP algorithms are trained to classify nine types of attacks on the cloud side of an IoT network. The TON_IoT dataset is divided into train, test, and validation datasets. The training dataset is 60% of the total dataset, and the test and validation datasets are each 20% of the total dataset.

4.1.1. LSTM-Based IDS on Cloud without FS Method

We studied different LSTM models with varying numbers of layers and neurons to find the optimal LSTM model for deployment on the cloud side of an IoT network for an IDS. The details of the LSTM model are shown in Table 3. To determine the optimal learning rate for training, Model1 and Model2 are trained using learning rates of 0.1, 0.01, and 0.001 with 50, 80, and 100 epochs. The results indicate that the best learning rate for training the LSTM model is 0.001. Therefore, Model3 and Model4 are trained using a learning rate of 0.001 with 50 epochs. The training and testing results of Model1–Model4 are shown in Table 3. The results show that Model 1, trained with a learning rate of 0.001 and 50 epochs, gives the detection accuracy of 94.2%.

Table 3. Test results of the LSTM and Conv1d models for IDS implementation on cloud.

ML Models	Models	Configuration Details	Epochs	Learning Rate	Accuracy	Precision	Recall	F1-Score
LSTM	Model1	Layers = 2, Layer1,2 = 100 neurons	50	0.001	94.2%	0.83	0.82	0.82
LSTM	Model2	Layers = 2, Layer1 = 120 neurons, Layer2 = 100 neurons	50	0.001	94.01%	0.83	0.83	0.83
LSTM	Model3	Layers = 2, Layer1 = 180 neurons, Layer2 = 140 neurons	50	0.001	94.02%	0.84	0.82	0.82
LSTM	Model4	Layers = 3, Layer1 = 180 neurons, Layer2 = 140 neurons, Layer3 = 100 neurons	50	0.001	93.87%	0.84	0.80	0.81
Conv1d	Model1	Layers = 2, Kernel size = 3, Filters = 64,32	50	0.001	79.8%	0.59	0.59	0.57
Conv1d	Model2	Layers = 3, Kernel size = 3, Filters = 64, 32 , Layer3 = 100 neurons	50	0.001	73%	0.41	0.41	0.40
Conv1d	Model3	Layers = 1, Kernel size = 3, Filters = 32	50	0.001	76.5%	0.59	0.49	0.50
Conv1d	Model4	Layers = 1, Kernel size = 5, Filters = 32	50	0.001	79%	0.48	0.52	0.45
Conv1d	Model5	Layers = 1, Kernel size = 5, Filters = 64	50	0.001	80%	0.55	0.53	0.515
Conv1d	Model6	Layers = 1, Kernel size = 7, Filters = 64	50	0.001	83.28%	0.65	0.60	0.59

4.1.2. Conv1d-Based IDS on Cloud without FS Method

We studied different models of Conv1d for IDS implementation on the cloud. The Conv1d results are reported for different models after changing their number of layers, filters, and kernel sizes. All models are trained for 50 epochs with a learning rate of 0.001, after observing the best results with this learning rate. The details of the models are mentioned in Table 3. The testing and training results of Conv1D for IDS implementation on the cloud are shown in Table 3. The results show that the single-layered Model6, with a kernel size of 7 and 64 filters, outperformed the other models, achieving an accuracy of 83.28%, a precision of 0.65, a recall of 0.60, and an F1-score of 0.59. However, this accuracy of 83.28% is not sufficient for the implementation of IDS due to the low precision, F1-score, and recall.

4.1.3. Random Forest-Based IDS Implementation on Cloud without FS Methods

The random forest model is trained on the TON_IoT dataset with 50, 100, 150, and 200 estimators. The testing results of the RF model are shown in Table 4. The results show that there is a nominal difference in the performance of different RF models. It is observed

that Model2 showed the best results with a testing accuracy of 99.4%, precision of 0.95, recall of 0.97, and F1-score of 0.96.

Table 4. Test results of RF, XGBoost, and MLP models evaluated for IDS implementation on cloud.

ML Models	Models	No. of Estimators/Neurons	Accuracy	Precision	Recall	F1-Score
RF	Model1	50	99.40%	0.95	0.97	0.96
RF	Model2	100	99.30%	0.95	0.97	0.96
RF	Model3	150	99.38%	0.95	0.97	0.96
RF	Model4	200	99.41%	0.95	0.97	0.96
XGBoost	Model1	50	99.46%	0.96	0.97	0.96
XGBoost	Model2	100	99.49%	0.96	0.97	0.96
XGBoost	Model3	150	99.50%	0.96	0.97	0.97
XGBoost	Model4	200	99.50%	0.96	0.97	0.97
MLP	Model1	50	93.40%	0.83	0.80	0.80
MLP	Model2	100	93.70%	0.83	0.81	0.80
MLP	Model3	150	93.90%	0.84	0.81	0.82
MLP	Model4	200	93.87%	0.84	0.81	0.82

4.1.4. XGBoost-Based IDS Implementation on Cloud without FS Methods

The XGBoost model is trained on the TON_IoT dataset with 50, 100, 150, and 200 estimators. The testing results of the XGBoost-based IDS are shown in Table 4. The results show that there is a nominal difference in the performance of the XGBoost model for 50, 100, 150, and 200 estimators. It is observed that the XGBoost model with 150 estimators showed the best results with a testing accuracy of 99.50%, precision of 0.96, recall of 0.97, and F1-score of 0.97.

4.1.5. MLP-Based IDS Implementation on Cloud without FS Methods

Four models of MLP are tested for IDS implementation on the cloud without using FS techniques. Model1 had 50 neurons in the hidden layer, Model2 had 150 neurons in the hidden layer, Model3 had 250 neurons in the hidden layer, and Model4 had 400 neurons in the hidden layer. The results of the MLP models are described in Table 4. The results indicate that MLP Model3 outperformed the other MLP models with an accuracy of 93.90%, a precision of 0.84, a recall of 0.81, and an F1-score of 0.82.

4.1.6. Comparison of Machine Learning-Based IDS on the Cloud without Using FS Techniques

We deployed and compared the LSTM, Conv1d, RF, MLP, and XGBoost algorithms for IDS implementation on the cloud. From the results, it is evident that XGBoost presented the best results with a detection accuracy of 99.50%. After XGBoost, the second-best results are presented by the RF algorithm with a detection accuracy of 99.4% on attacks. The third-best results are presented by the LSTM algorithm with a detection accuracy of 94.2% followed by MLP with 93.9%. Overall XGBoost and RF are suitable for the implementation of an IDS in terms of accuracy, and Conv1d is not feasible for the implementation of an IDS on the cloud.

4.2. Machine Learning-Based IDS on the Cloud with FS Methods

The FS techniques are applied to improve the accuracy of the IDS. Chi-square and PCA techniques are used for FS on the best models of XGBoost, LSTM, RF, and MLP.

4.2.1. LSTM-Based IDS on the Cloud with FS/FE Methods

The performances of LSTM with 25, 15, and 10 features are presented in Table 5. The testing accuracies with PCA are 94.07%, 94.4%, and 94.4% for 25, 15, and 10 extracted features, respectively. In contrast, the chi-square demonstrates testing accuracies of 94.25%, 94.5%, and 94% for 25, 15, and 10 selected features, respectively. The results reveal that with the chi-square FS technique, the accuracy of LSTM improved using 15 features compared to not employing any FS technique.

Table 5. Test results of LSTM-, RF-, MLP-, XGBoost-based IDSs with FS methods.

ML Model	FS Technique	No. of Features	Accuracy	Precision	Recall	F1-Score
LSTM	PCA	25	94.07%	0.82	0.84	0.83
LSTM	Chi-square	25	94.25%	0.84	0.84	0.84
LSTM	PCA	15	94.4%	0.84	0.83	0.83
LSTM	Chi-square	15	94.5%	0.84	0.83	0.83
LSTM	PCA	10	94.4%	0.84	0.84	0.84
LSTM	Chi-square	10	94.0%	0.84	0.82	0.83
RF	PCA	25	97.50%	0.92	0.91	0.91
RF	Chi-square	25	96.50%	0.90	0.87	0.88
RF	PCA	15	94.64%	0.85	0.84	0.84
RF	Chi-square	15	94.40%	0.84	0.84	0.84
RF	PCA	10	94.44%	0.84	0.84	0.84
RF	Chi-square	10	94.53%	0.85	0.82	0.85
MLP	PCA	25	93.90%	0.84	0.82	0.82
MLP	Chi-square	25	93.88%	0.84	0.81	0.81
MLP	PCA	15	94.00%	0.84	0.82	0.83
MLP	Chi-square	15	93.83%	0.83	0.80	0.80
MLP	PCA	10	93.99%	0.84	0.81	0.82
MLP	Chi-square	10	92.99%	0.82	0.81	0.81
XGBoost	PCA	25	97.67%	0.98	0.98	0.98
XGBoost	Chi-square	25	98.35%	0.98	0.98	0.98
XGBoost	PCA	15	95.58%	0.96	0.96	0.96
XGBoost	Chi-square	15	98.13%	0.98	0.98	0.98
XGBoost	PCA	10	95.55%	0.96	0.96	0.96
XGBoost	Chi-square	10	95.69%	0.96	0.96	0.96

4.2.2. RF-Based IDS on the Cloud with FS Methods

PCA and chi-square are deployed on the RF algorithm to record the results. The testing accuracies of the RF algorithm on PCA are 97.5%, 94.64%, and 94.44% for 25, 15, and 10 features, respectively. On chi-square, the accuracies for 25, 15, and 10 selected features are 96.5%, 94.4%, and 94.53%, respectively. The detailed results are presented in Table 5.

4.2.3. MLP-Based IDS Implementation on the Cloud with FS Methods

PCA and chi-square are deployed on the MLP algorithm, and their results are recorded for 25, 15, and 10 features, respectively. On PCA, MLP testing accuracies are 93.9%, 94.0%, and 93.99% for 25, 15, and 10 extracted features, respectively. On chi-square, MLP testing accuracies are 93.88%, 93.83%, and 92.99% for 25, 15, and 10 features, respectively. The results are presented in Table 5.

4.2.4. XGBoost-Based IDS Implementation on the Cloud with FS Methods

The testing results of XGBoost trained with 10, 15, and 25 features selected/extracted by chi-square and PCA are shown in Table 5. The accuracy of XGBoost with 25 features computed by PCA is 97.67%, while for features selected with chi-square, it is 98.35%. Similarly, for 15 features computed by PCA, the accuracy is 95.58%, while with chi-square, the accuracy is 98.13%. Finally, with 10 features computed by PCA, the accuracy is 95.55%, and with chi-square, the accuracy is 95.69%. In terms of accuracy, XGBoost trained with 25 features selected by chi-square achieved the highest accuracy of 98.35%.

4.2.5. Comparison of XGBOOST-, LSTM-, RF-, and MLP-Based IDS on the Cloud with FS Techniques

FS involves choosing a subset of the most informative features from the original set while reducing dimensionality. It aims to retain the semantic interpretability of the selected features. In contrast, FE transforms the original features into a new low-dimensional space using mathematical projection. Although it effectively reduces dimensionality, the extracted features lose their intuitive meanings.

For an IDS, FS allows the creation of a lightweight and efficient IDS by carefully selecting relevant original features. On the other hand, FE techniques provide a valuable way to transform and distill the essence of the original feature set, reducing overall data dimensionality while preserving critical information. Both FS and FE are essential tools for enhancing the cybersecurity posture of IoT ecosystems, ensuring effective threat detection tailored to the limitations and intricacies of IoT devices and networks.

Two FS/FE techniques, i.e., PCA and chi-square, are applied to the XGBoost-, LSTM-, RF-, and MLP-based IDSs. The results for LSTM showed that chi-square performed best with 15 out of 40 selected features, achieving a detection accuracy of 94.5%. With RF, PCA proved to be the most effective technique, resulting in a detection accuracy of 97.5% using 25 features. Similarly, for MLP, PCA resulted in the best detection accuracy of 94% with 15 features. XGBoost trained with features selected by chi-square gave the best results. With 25 features, the accuracy is 98.35%, while with 15 features the accuracy is 98.13%. The results above indicate that the highest training accuracy attained with the FS/FE technique is 98.35%, whereas without using FS/FE, it reaches 99.5%. While reducing features might reduce prediction time for an IDS, prioritizing accuracy remains crucial for effective IDS implementation, even if it means retaining a higher number of features. Therefore, the XGBoost-based IDS without FS/FE outperformed the other ML-based IDSs evaluated in this study.

We compared the inference times of the FS and FE techniques. We analyzed the inference times of the XGBoost models trained with both FS and FE as shown in Table 6. The simulations are carried out on Intel(R) Core(TM) i5-8250U CPU with 8GB RAM. The inference time of XGBoost without FS/FE is 0.6631 milliseconds. The chi-square method (FS) showed a higher inference time compared to the PCA method (FE). Overall, the inference time with FS/FE was more than double that of the XGBoost model without them.

While FS/FE reduced the inference time of XGBoost compared to no FS, real-time testing considers the combined time of both FS/FE and XGBoost inference. Importantly, FS is a one-time calculation performed during training. In the testing phase, directly selecting the top features from the high-dimensional data for dimensionality reduction can be a preferred choice. However, applying FS to real-time captured packets incurs a higher time cost than XGBoost inference itself.

Our discussions and results align with the detailed research presented in [42,43]. For time-critical attacks like DoS/DDoS and MitM, a low inference time is crucial for the IDS. Therefore, we deployed the XGBoost model (estimator = 150) without FS/FE due to its superior accuracy and faster inference time.

Table 6. PCA vs chi-square for multiclassification using XGBoost—*inference time in milliseconds*.

No. of Features	PCA	XGBoost + PCA	Chi-Square	XGBoost + Chi-Square
10	1.1547	1.7944	1.3518	1.9761
15	1.2732	1.8900	1.4709	2.0856
25	1.2629	1.8813	1.6112	2.2363

4.3. Machine Learning-Based IDS Implementation on IoT Device

An ML-based IDS has been embedded in a microcontroller-based smart thermostat, a low-power and cost-efficient IoT device functioning as an HTTP client to upload building temperature and humidity data to the AWS cloud server via WiFi technology. The RF-, XGBoost-, DT-, and ANN-based IDSs are embedded in the smart thermostat to determine their feasibility in terms of accuracy, storage requirements, and inference time. The TinyML library is used to embed the trained IDS on the microcontroller. The trained models of RF, XGBoost, ANN, and DT are embedded in the smart thermostat to classify the network traffic as attack or normal. Low-power IoT devices are vulnerable to various security threats due to their limited resources. Consequently, these devices may be exploited for backdoor, DoS/DDoS, MITM, scanning, and ransomware attacks. Given the restricted access of low-cost IoT devices to network traffic features, researchers opted to deploy an IDS on a Raspberry Pi-based edge device, leveraging tools like tcpdump and Wireshark for extracting network features. This study focuses on implementing the primary defense layer on the microcontroller-based IoT device (smart thermostat), which lacks packet analyzer tools. An ML-based IDS is specifically designed for the binary classification of attacks. The IDS classified backdoor, scanning, DDoS, MITM, and ransomware as attacks (i.e., 1), while normal traffic was classified as 0.

4.3.1. ML-Based IDS Implementation on Smart Thermostat Using Connection, Statistical, and DNS Activity Features of TON_IoT Dataset

In this work, ANN, RF, XGBoost, and DT are tested as ways of implementing the binary classifier IDS. The TON_IoT dataset is used for training the IDS at the edge device. The dataset is split into 70% training and 30% testing data. The 10-fold cross-validation has been applied during training to ensure the generalization of the ML model. Due to limited computational and storage resources at the edge device, we trained the IDS with 25 features (i.e., connection activity, statistical activity, and DNS activity). The ML-based IDS is evaluated in terms of accuracy, precision, recall, F1-score, false positive rate (FPR), and false negative rate (FNR). The testing results of the binary classification of the XGBoost-, RF-, DT-, and ANN-based IDSs are shown in Table 7. These results indicate that the RF-based IDS (100 estimators, depth = 8) outperformed the other RF models. Additionally, the DT model with a depth of 18 achieved an accuracy of 99.85%. Regarding ANN, experiments were conducted with 50, 100, 150, and 200 neurons, revealing that all ANN models performed best at 100 epochs. Among these, the three-layered ANN model exhibited the highest accuracy of 98.93%. XGBoost outperformed the other ML models and achieved the highest accuracy of 99.92%.

The implementation results are shown in Table 7. The table shows the testing accuracy, program storage, and inference time average of 30 packets. DT outperformed RF, XGBoost, and ANN in terms of inference time, being capable of classifying network traffic as normal/attack in 5.09 μ s with 99.05% accuracy. The DT-based IDS (depth = 8) consumed 65.94% of program memory of the IoT device. The DT model with a depth of 20 achieved the best accuracy of 99.85% and the inference time for classifying network traffic is 11.36 μ s. The precision, recall, and F1-score for this model are 0.99, while the program storage required for this model is 66.73%. In terms of accuracy, FNR, FPR, and XGBoost with 100 estimators outperformed RF, ANN, and DT with an accuracy of 99.92%, FPR of 0.04%, and FNR of 0.23%, but the inference time is 3002.78 μ s. The best accuracy achieved by the RF-based IDS is 98.89% with 100 estimators. The FNR for this model is 1.28% and the FPR is 1.06%. It

is not feasible to implement the RF model with 200 estimators since the required memory is not available on an ESP32-based edge device. The ANN model-based IDS is also studied in this work. Two-layered and three-layered ANN models are used for the IDS. The best accuracy achieved with the ANN model (Layer 1 = 96, Layer 2 = 64, Layer 3 = 32) is 98.93%.

Table 7. IDS implementation on microcontroller-based IoT device using ToN_IoT dataset (connection, statistical, and DNS activity features).

ML Model (No. of Estimators)	Accuracy	Precision	Recall	F1-Score	FNR	FPR	Computation Time μ s (Average)	Program Storage
XGBoost (Estimator = 50, Depth = 8)	99.91%	0.99	0.99	0.99	0.23%	0.04%	1840.70	80.92%
XGBoost (Estimator = 100, Depth = 8)	99.92%	0.99	0.99	0.99	0.23%	0.04%	3002.78	82.71%
XGBoost (Estimator = 150, Depth = 8)	99.92%	0.99	0.99	0.99	0.23%	0.04%	3514.39	83.34%
XGBoost (Estimator = 200, Depth = 8)	99.91%	0.99	0.99	0.99	0.25%	0.04%	4100.54	83.86%
RF (Estimator = 50, Depth = 8)	98.67%	0.95	0.98	0.97	1.34%	1.32%	628.62	72.76%
RF (Estimator = 100, Depth = 8)	98.89%	0.96	0.98	0.97	1.28%	1.06%	1565.01	79.0%
RF (Estimator = 150, Depth = 8)	98.72%	0.95	0.98	0.97	1.28%	1.28%	2679.09	85.12%
RF (Estimator = 200, Depth = 8)	98.71%	0.95	0.98	0.97	1.28%	1.29%	—	Memory overflow
DT (Depth = 6)	97.94%	0.92	0.99	0.95	1.01%	2.34%	6.05	65.83%
DT (Depth = 8)	99.05%	0.98	0.97	0.97	2.37%	0.56%	5.09	65.94%
DT (Depth = 10)	99.53%	0.98	0.99	0.99	0.92%	0.34%	6.96	66.13%
DT (Depth = 12)	99.69%	0.99	0.99	0.99	0.54%	0.25%	8.82	66.40%
DT (Depth = 14)	99.78%	0.99	0.99	0.99	0.45%	0.15%	9.90	66.56%
DT (Depth = 16)	99.84%	0.99	0.99	0.99	0.37%	0.10%	11.61	66.68%
DT (Depth = 18)	99.85%	0.99	0.99	0.99	0.34%	0.10%	11.82	66.72%
DT (Depth = 20)	99.85%	0.99	0.99	0.99	0.31%	0.10%	11.36	66.73%
ANN (Layer 1 = 32, Layer 2 = 16)	98.62%	0.97	0.96	0.97	3.31%	0.65%	131.98	75%
ANN (Layer 1 = 64, Layer 2 = 32)	98.98%	0.97	0.97	0.97	2.59%	0.58%	296.12	76.15%
ANN (Layer 1 = 64, Layer 2 = 32, Layer 3 = 16)	98.92%	0.97	0.97	0.97	2.74%	0.62%	350.16	76.34%
ANN (Layer 1 = 96, Layer 2 = 64, Layer 3 = 32)	98.93%	0.97	0.97	0.97	2.41%	0.70%	3088.97	78.32%

Due to the limited storage and computational power of the microcontroller-based edge device, our objective is to identify an IDS that can deliver accurate results within a shorter time frame. The XGBoost-based IDS has proven superior to DT, RF, and ANN in terms of accuracy, FNR, and FPR. The highest accuracy achieved by the XGBoost-based IDS is 99.92%, but it requires 3002.78 μ s for threat detection. In contrast, XGBoost's performance is only 0.07% better than DT, yet DT completes computations in 11.36 μ s. Considering DT's significantly quicker detection time compared to XGBoost, it could be the preferred choice

for implementing the IDS on edge devices. However, the accuracy of the IDS is crucial. Therefore, XGBoost should be preferred for the implementation of the IDS.

4.3.2. ML-Based IDS Implementation on Smart Thermostat Using Connection and Statistical Activity Features

Considering the limited access to the lwIP library and the nature of our application, only connection activity and statistical activity network features are considered. To avoid overfitting, we excluded timestamp, source and destination IP addresses. The input features that can be extracted on the Esp32-based smart thermostat are source port, destination port, proto, service, duration, source bytes, destination bytes, connection state, missed bytes, source packets, source IP header bytes, destination packets, and destination IP header bytes. Based on these 13 features, the XGBoost, RF, DT, and ANN models are trained for binary classification and the results are shown in Table 8. The results show that XGBoost with the depth of 8 and 100 estimators outperformed DT, RF, and ANN in terms of accuracy, FPR, and FNR, while DT with a depth of 5 is the fastest in terms of inference time. The best results of XGBoost, DT, RF, and ANN are summarized in Figure 3. XGBoost (100 estimators, depth = 8), DT depth = 5), RF (150 estimators, depth = 8), and ANN are compared in terms of accuracy, program memory, inference time, FPR, and FNR. The F1-score, recall, precision, and accuracy are summarized in the subfigure titled Accuracy(%). Although the inference time of DT is 3.2 μ s, the FPR and FNR of DT are 0.60% and 0.72%, respectively, compared to XGBoost, which has an FPR and FNR of 0.30% and 0.06%, respectively. Since the required program memory is available in the microcontroller of the smart thermostat, XGBoost is the preferred choice for implementing an IDS with 13 features due to its superior performance.

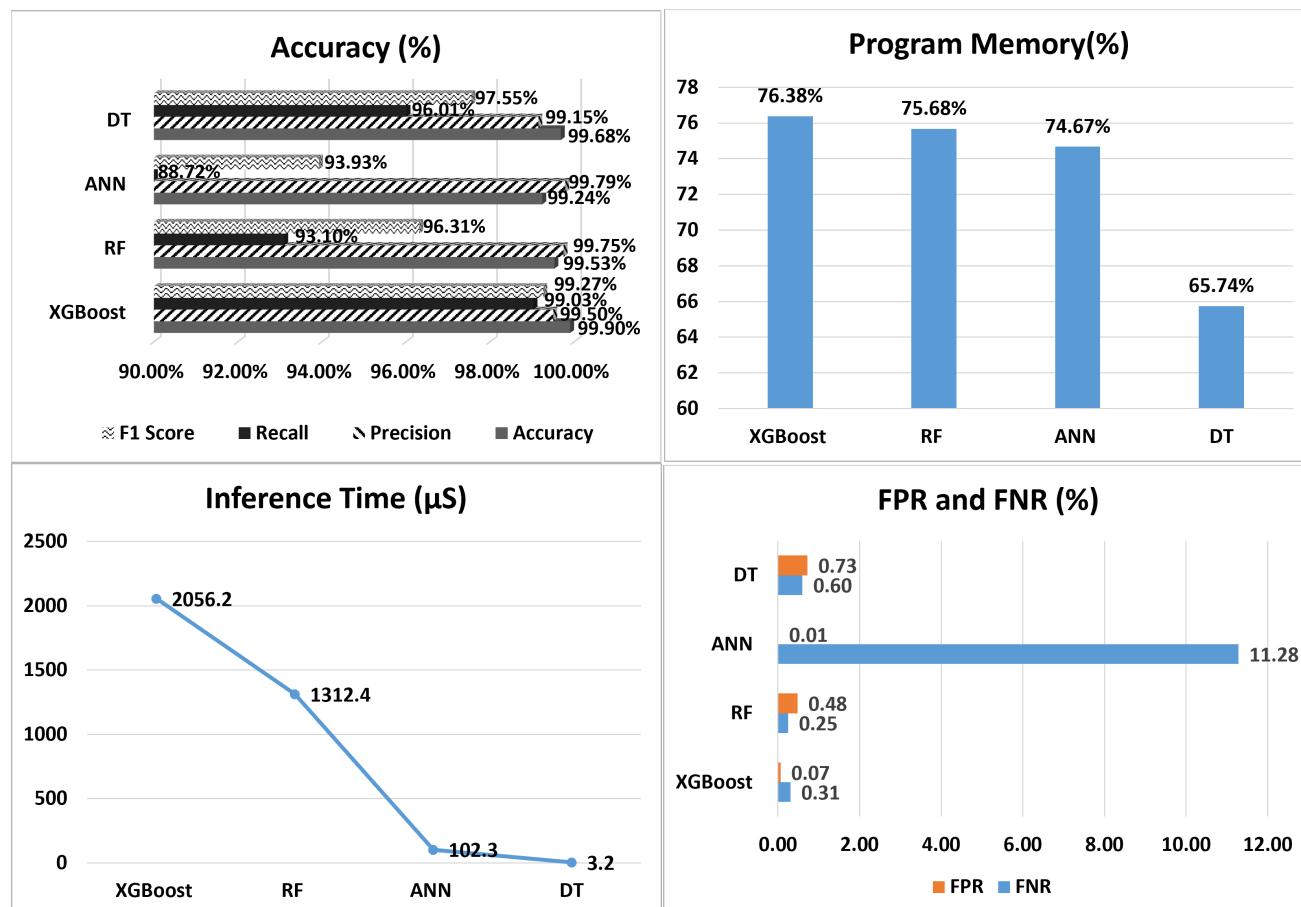


Figure 3. Comparison of XGBoost-, RF-, DT-, and ANN-based IDS implementation on smart thermostat using Ton_IoT dataset.

Table 8. IDS implementation on edge using connection and statistical activity features of TON_IoT dataset.

ML Model (No. of Estimators)	Accuracy	Precision	Recall	F1-Score	FNR	FPR	Computation Time μ s (Average)	Program Storage
XGBoost (Estimator = 50, Depth = 8)	99.92%	0.99	0.99	0.99	0.35%	0.05%	1223.4	75.0%
XGBoost (Estimator = 100, Depth = 8)	99.91%	0.99	0.99	0.99	0.30%	0.06%	2056.2	76.38%
XGBoost (Estimator = 150, Depth = 8)	99.92%	0.99	0.99	0.99	0.30%	0.06%	2480	76.8%
XGBoost (Estimator = 200, Depth = 8)	99.92%	0.99	0.99	0.99	0.30%	0.06%	2936.7	77.32%
RF (Estimator = 50, Depth = 8)	99.52%	0.99	0.92	0.96	0.25%	0.49%	407.9	70.57%
RF (Estimator = 100, Depth = 8)	99.52%	0.99	0.93	0.96	0.25%	0.48%	1312.4	75.68%
RF (Estimator = 150, Depth = 8)	99.53%	0.99	0.93	0.96	0.25%	0.47%	2729.7	80.39%
RF (Estimator = 200, Depth = 8)	99.52%	0.99	0.93	0.96	0.25%	0.48%	3584.2	85.08%
DT (Depth = 5)	99.68%	0.99	0.96	0.97	0.60%	0.72%	3.2	65.74%
DT (Depth = 10)	99.59%	0.99	0.94	0.96	0.79%	0.38%	8.4	65.96%
DT (Depth = 15)	99.58%	0.99	0.94	0.96	0.93%	0.38%	12	66.23%
DT (Depth = 20)	99.59%	0.99	0.94	0.96	0.84%	0.30%	11.4	66.34%
ANN (Layer 1 = 32, Layer 2 = 16)	99.24%	0.99	0.88	0.93	11.28%	0.01%	102.3	74.67%
ANN (Layer 1 = 64, Layer 2 = 32)	99.23%	0.99	0.88	0.97	11.09%	0.01%	242.8	75.87%
ANN (Layer 1 = 64, Layer 2 = 32, Layer 3 = 16)	99.17%	0.98	0.89	0.93	10.78%	0.12%	478.2	76.01%
ANN (Layer 1 = 96, Layer 2 = 64, Layer 3 = 32)	99.22%	0.99	0.89	0.93	10.83%	0.06%	3986.2	76.15%

4.4. Real-Time Testing of IDS on Smart Thermostat

The smart thermostat uploads the data on the AWS cloud server (<http://16.170.2.15.67/display>). The smart home application provides users with remote access to turn ON/OFF the HVAC through the webpage. This bidirectional communication is vulnerable to DoS and MITM attacks. Using our smart thermostat dataset, we trained XGBoost, DT, RF, and ANN for IDS implementation on the smart thermostat. The testing results of the IDS implemented with XGBoost, RF, DT, and ANN are shown in Table 9. The IDS implemented with XGBoost (depth = 8, estimator = 100) achieved the highest accuracy of 97.66%. The best accuracy achieved with RF (depth is 8, estimators = 20) is 95.33%, with DT (with a depth of 15) it is 96.54%, and with ANN it is 93.41% as shown in Figure 4. The accuracy, F1-score, recall, and precision are summarized under the subfigure titled Accuracy (%), whereas the comparison of program memory requirements, inference time, FPR, and FNR are also shown in Figure 4. The results clearly show that although the inference time of XGBoost is high and it requires more program memory, it outperformed DT, ANN, and RF in terms of accuracy, F1-score, recall, precision, FPR, and FNR.

Table 9. IDS implementation on smart thermostat using IDSH dataset.

ML Model (No. of Estimators)	Accuracy	Precision	Recall	F1-Score	FNR	FPR	Computation Time μs (Average)	Program Storage
XGBoost (Estimator = 50, Depth = 8)	97.59%	0.97	0.97	0.97	2.76%	2.24%	1502.3	77.0%
XGBoost (Estimator = 100, Depth = 8)	97.66%	0.97	0.97	0.97	2.76%	2.12%	3515.9	84.01%
XGBoost (Estimator = 150, Depth = 8)	97.50%	0.97	0.97	0.97	3.26%	2.12%	Memory Out of Bound	–
XGBoost (Estimator = 200, Depth = 8)	97.58%	0.97	0.97	0.97	3.01%	2.12%	Memory Out of Bound	–
RF (Estimator = 50, Depth = 8)	95.17%	0.93	0.96	0.94	1.19%	2.48%	439	69.53%
RF (Estimator = 100, Depth = 8)	95.17%	0.93	0.96	0.94	1.00%	6.61%	1357.6	73.46%
RF (Estimator = 150, Depth = 8)	95.25%	0.93	0.96	0.95	1.00%	6.50%	2331.4	77.29%
RF (Estimator = 200, Depth = 8)	95.33%	0.93	0.96	0.95	1.00%	6.38%	3314.4	80.96%
DT (Depth = 5)	90.59%	0.88	0.90	0.89	8.54%	9.81%	3.7	66.54%
DT (Depth = 10)	94.61%	0.93	0.94	0.94	4.77%	5.67%	6.2	66.61%
DT (Depth = 15)	96.54%	0.96	0.96	0.96	5.77%	2.36%	7.3	66.71%
DT (Depth = 20)	96.38%	0.96	0.95	0.96	6.03%	2.48%	8.0	66.78%
ANN (Layer 1 = 32, Layer 2 = 16)	92.76%	0.85	0.93	0.89	6.28%	7.68%	117.5	78.02%
ANN (Layer 1 = 64, Layer 2 = 32)	91.23%	0.80	0.97	0.87	3.01%	11.46%	286.6	76.11%
ANN (Layer 1 = 64, Layer 2 = 32, Layer 3 = 16)	92.92%	0.84	0.95	0.89	4.77%	8.15%	313.9	75.92%
ANN (Layer 1 = 96, Layer 2 = 64, Layer 3 = 32)	93.41%	0.85	0.96	0.90	3.26%	8.15%	2498.1	75.28%

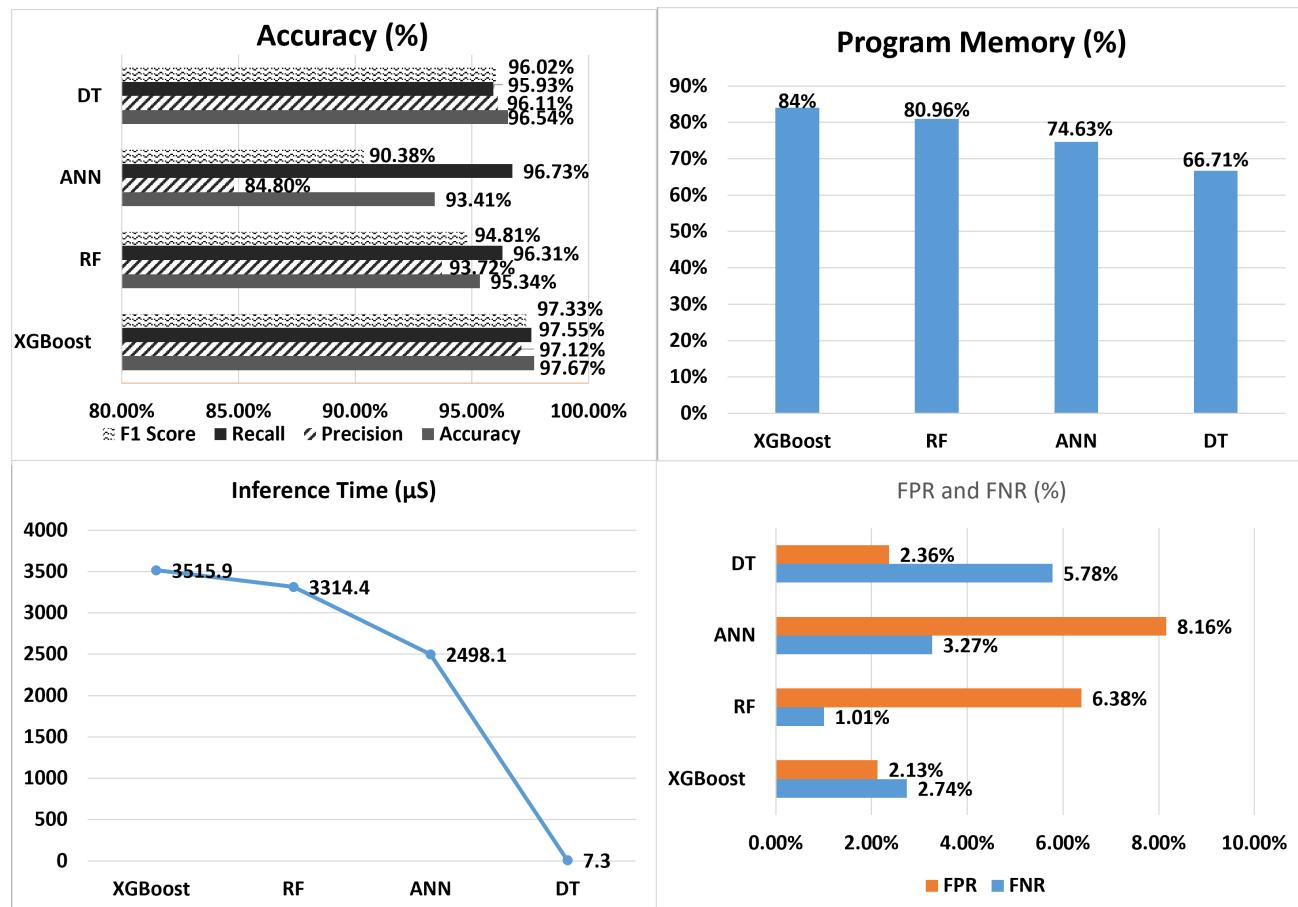


Figure 4. Comparison of XGBoost-, RF-, DT-, and ANN-based IDS implementation on smart thermostat using IDSH dataset.

4.5. Discussion

In this study, the IDS is deployed on a smart thermostat built with the ESP32 microcontroller. The microcontroller boasts 440 KB of ROM and 520 KB of SRAM for program memory and instructions, respectively.

We conducted an initial feasibility test by implementing the IDS, incorporating connection activity, statistical activity, and DNS activity features. Utilizing XGBoost, the IDS consumed 82.71% of program memory and required 3002.78 μ s to compute the output. Conversely, the IDS implemented with RF consumed 79.0% of program memory and took 1565.01 μ s for computation. Similarly, the IDS implemented with ANN utilized 76.15% of program memory and took 296.12 μ s for computation. Notably, DT outperformed XGBoost, ANN, and RF in terms of program memory and inference time, consuming just 66.73% of program memory and requiring only 11.37 μ s for computation.

We also implemented the IDS using connection and statistical activity features. The most efficient XGBoost model utilized 76.38% of program storage and took 2056.2 μ s for computation. Following closely in accuracy, the DT model consumed 65.74% of program memory with an inference time of only 3.2 μ s. In contrast, RF consumed 75.68% of program memory with an inference time of 1312.4 μ s. The ANN model, ranking fourth in accuracy, consumed 75.87% of program memory with an inference time of 242.8 μ s.

Furthermore, we implemented the IDS with the IDSH dataset, which combines sensor and network datasets. The highest accuracy achieved by XGBoost (estimator = 100, depth = 8) is 97.66% but with an inference time of 3515.9 μ s, whereas with XGBoost (estimator = 50, depth = 8), the IDS consumed 77.0% of program memory with an inference time of 1502.3 μ s. The RF model consumed 80.96% of program memory with an inference time of 3314.4 μ s, while the DT model consumed 66.78% of program memory with an inference time of 8.0 mi-

croseconds. The ANN model consumed 75.28% of program memory with an inference time of 2498.1 μ s. It is noteworthy that for the IDSH dataset, the XGBoost models with estimators of 150 and above encountered memory constraints preventing their implementation on the smart thermostat.

DT operates on a simpler model, making decisions based on if-else rules at each node, which requires less computation time compared to the matrix multiplications and activity functions utilized in XGBOOST, ANN, and RF. As a result, the inference time of DT is less than that of XGBOOST, ANN, and RF. However, XGBoost captures complex relations in the data and handles non-linear patterns more effectively. Moreover, it possesses the capability to consider both past and future covariates present in the input data. The internal feature of hyperparameter tuning further contributes to its higher accuracy, making XGBoost's accuracy superior.

Table 9 and Figure 4 present the results of the IDS deployed on the smart thermostat device using our IDSH dataset. The trained ML models were embedded in the ESP32 microcontroller-based smart thermostat, and the inference time was calculated after generating attacks from our Raspberry Pi-based attacker node. Based on these results, we determined that the inference time of XGBoost exceeds that of DT. However, given that the building dynamics are slow and a delay of a few milliseconds in HVAC control input will not significantly impact the indoor environment, we prioritize accuracy. Therefore, for the smart thermostat, the XGBoost (estimator = 50, depth = 8)-based IDS is embedded in the smart thermostat for intrusion detection.

5. Conclusions and Future Work

In this work, a two-layered ML-based IDS is proposed. The proposed IDS is trained to be deployed on IoT devices and the cloud side of an IoT network to detect threats as soon as possible. The first layer of the IDS is embedded in the microcontroller-based smart thermostat using TinyML for binary classification. DDoS, MITM, ransomware, scanning, and backdoor attacks are detected on IoT devices. These attacks are labeled as an attack class for binary classification. The results show that the decision tree (DT)-based IDS can detect the attacks with 99.85% accuracy in 11.36 μ s. The smart thermostat is also tested against real attacks (DoS, MITM) generated by the Raspberry Pi-based attacker node, and the accuracy of XGBoost is 97.59%. The dataset is comprised of 4144 samples, which were collected in this work.

The second layer of the IDS is deployed on the cloud, and five ML algorithms, namely LSTM, RF, MLP, Conv1d, and XGBoost, are trained to detect attacks such as DoS, DDoS, port scanning, MITM, XSS/injection, brute force, backdoor, and ransomware. The XGBoost algorithm showed the best results on the cloud with a detection accuracy of 99.5%, followed by RF, which showed the second-best results on the TON_IoT dataset with a detection accuracy of 99.4%. Two FS techniques, chi-square and PCA, are also employed on algorithms trained on the cloud side of the IoT network, but they did not significantly improve detection accuracy. The best results are obtained with chi-square, which showed 98.35% accuracy with XGBoost, while the best accuracy achieved with PCA is 97.67% with XGBoost. The IDS on the cloud detects incoming traffic and, if it is classified as normal, uploads it to the webpage.

Based on the experimental results, it is found that the DT-based IDS outperformed ANN, RF, and XGBoost in terms of inference time and memory requirements, while the XGBoost-based IDS outperformed the other ML models in terms of accuracy, precision, recall, f1 score, FPR, and FNR. For an IDS, accuracy is crucial. Therefore, prioritizing accuracy over inference time, XGBoost is the preferred choice for IDS implementation on the cloud and smart thermostat due to its superior performance in terms of accuracy. The XGBoost-based IDS is deployed on the cloud, and prediction results are also shared on the website along with the smart thermostat data. For the future prospects of this work, the presented two-layered IDS will be tested for additional attacks, and we will extend the number of samples in our smart home dataset. Due to limited resources and the restricted

access of the lwIP library to network parameters, we managed to retrieve only a limited set of network parameters on the microcontroller. In the future, we will work on extracting additional network parameters at the IoT device.

Author Contributions: All authors contributed extensively to the work presented in this paper. Conceptualization, A.J., A.-u.-H.Q., and H.L.; methodology, A.J., A.E., M.J. and M.N.A.; software, A.J., A.E. and M.J.; validation, A.J., A.E., M.J., M.N.A., A.-u.-H.Q. and H.L.; formal analysis, A.J., H.L., M.N.A., M.J. and A.-u.-H.Q.; investigation, A.J., M.N.A., H.L. and A.-u.-H.Q.; resources, A.J., A.E. and A.-u.-H.Q.; data curation, A.J., A.E., M.N.A., H.L. and M.J.; writing—original draft preparation, A.J., A.E., M.J. and A.-u.-H.Q.; writing—review and editing, A.J., A.E., M.J., M.N.A., H.L. and A.-u.-H.Q.; visualization, A.J., M.N.A. and A.-u.-H.Q.; supervision, A.J. and A.-u.-H.Q.; project administration, A.J. and A.-u.-H.Q.; funding acquisition, A.J. and A.-u.-H.Q. All authors have read and agreed to the published version of the manuscript.

Funding: The APC was funded by Department of Cyber Security and Networks, School of Computing, Engineering and Built Environment, Glasgow Caledonian University, UK

Data Availability Statement: <https://www.kaggle.com/datasets/bc7c97914edab17a5821f5b27f465904bad79132f07cfcefc83e9669daeb3f98>(accessed on 26 April 2024.) Shorturl <http://surl.li/psyjo> (accessed on 26 April 2024.)

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. IoT.Business.News. Number of Connected IOT Devices Growing 9% to 12.3 Billion Globally, Cellular IOT Now Surpassing 2 Billion. Available online: <https://iotbusinessnews.com/2021/09/23/13465-number-of-connected-iot-devices-growing-9-to-12-3-billion-cellular-iot-now-surpassing-2-billion/> (accessed on 24 January 2024).
2. Hassija, V.; Chamola, V.; Saxena, V.; Jain, D.; Goyal, P.; Sikdar, B. A survey on IoT security: Application areas, security threats, and solution architectures. *IEEE Access* **2019**, *7*, 82721–82743. [CrossRef]
3. Singh, S.; Ra, I.H.; Meng, W.; Kaur, M.; Cho, G.H. SH-BlockCC: A secure and efficient Internet of things smart home architecture based on cloud computing and blockchain technology. *Int. J. Distrib. Sens. Netw.* **2019**, *15*, 1550147719844159. [CrossRef]
4. Sohail, S.; Fan, Z.; Gu, X.; Sabrina, F. Multi-tiered Artificial Neural Networks model for intrusion detection in smart homes. *Intell. Syst. Appl.* **2022**, *16*, 200152. [CrossRef]
5. Ferrag, M.A.; Maglaras, L.; Ahmim, A.; Derdour, M.; Janicke, H. Rdtids: Rules and decision tree-based intrusion detection system for internet-of-things networks. *Future Internet* **2020**, *12*, 44. [CrossRef]
6. Mighan, S.N.; Kahani, M. A novel scalable intrusion detection system based on deep learning. *Int. J. Inf. Secur.* **2021**, *20*, 387–403. [CrossRef]
7. Ahmed, S.; Khan, Z.A.; Mohsin, S.M.; Latif, S.; Aslam, S.; Mujlid, H.; Adil, M.; Najam, Z. Effective and efficient DDoS attack detection using deep learning algorithm, multi-layer perceptron. *Future Internet* **2023**, *15*, 76. [CrossRef]
8. Zhong, M.; Zhou, Y.; Chen, G. Sequential model based intrusion detection system for IoT servers using deep learning methods. *Sensors* **2021**, *21*, 1113. [CrossRef]
9. Gad, A.R.; Nashat, A.A.; Barkat, T.M. Intrusion detection system using machine learning for vehicular ad hoc networks based on ToN-IoT dataset. *IEEE Access* **2021**, *9*, 142206–142217. [CrossRef]
10. Meidan, Y.; Avraham, D.; Libhaber, H.; Shabtai, A. CADeSH: Collaborative anomaly detection for smart homes. *IEEE Internet Things J.* **2022**, *10*, 8514–8532. [CrossRef]
11. Shi, L.; Wu, L.; Guan, Z. Three-layer hybrid intrusion detection model for smart home malicious attacks. *Comput. Electr. Eng.* **2021**, *96*, 107536. [CrossRef]
12. Thakur, S.; Chakraborty, A.; De, R.; Kumar, N.; Sarkar, R. Intrusion detection in cyber-physical systems using a generic and domain specific deep autoencoder model. *Comput. Electr. Eng.* **2021**, *91*, 107044. [CrossRef]
13. Rahman, M.A.; Asyhari, A.T.; Leong, L.; Satrya, G.; Tao, M.H.; Zolklipli, M. Scalable machine learning-based intrusion detection system for IoT-enabled smart cities. *Sustain. Cities Soc.* **2020**, *61*, 102324. [CrossRef]
14. Reddy, D.K.K.; Behera, H.S.; Nayak, J.; Naik, B.; Ghosh, U.; Sharma, P.K. Exact greedy algorithm based split finding approach for intrusion detection in fog-enabled IoT environment. *J. Inf. Secur. Appl.* **2021**, *60*, 102866. [CrossRef]
15. Labiod, Y.; Amara Korba, A.; Ghoualmi, N. Fog computing-based intrusion detection architecture to protect iot networks. *Wirel. Pers. Commun.* **2022**, *125*, 231–259. [CrossRef]
16. Kumar, P.; Gupta, G.P.; Tripathi, R. Design of anomaly-based intrusion detection system using fog computing for IoT network. *Autom. Control Comput. Sci.* **2021**, *55*, 137–147. [CrossRef]
17. Pacheco, J.; Benitez, V.H.; Felix-Herran, L.C.; Satam, P. Artificial neural networks-based intrusion detection system for internet of things fog nodes. *IEEE Access* **2020**, *8*, 73907–73918. [CrossRef]

18. Hafeez, I.; Antikainen, M.; Ding, A.Y.; Tarkoma, S. IoT-KEEPER: Detecting malicious IoT network activity using online traffic analysis at the edge. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 45–59. [[CrossRef](#)]
19. Eskandari, M.; Janjua, Z.H.; Vecchio, M.; Antonelli, F. Passban IDS: An Intelligent Anomaly-Based Intrusion Detection System for IoT Edge Devices. *IEEE Internet Things J.* **2020**, *7*, 6882–6897. . [[CrossRef](#)]
20. Hosseinioorbin, S.; Layeghy, S.; Sarhan, M.; Jurdak, R.; Portmann, M. Exploring edge TPU for network intrusion detection in IoT. *J. Parallel Distrib. Comput.* **2023**, *179*, 104712. [[CrossRef](#)]
21. Bangui, H.; Buhnova, B. Lightweight intrusion detection for edge computing networks using deep forest and bio-inspired algorithms. *Comput. Electr. Eng.* **2022**, *100*, 107901. . [[CrossRef](#)]
22. Javed, A. Intrusion Detection in Smart Homes (IDSH) Dataset. Available online: <https://www.kaggle.com/datasets/bc7c97914edab17a5821f5b27f465904bad79132f07cfcefc83e9669daeb3f98> (accessed on 24 January 2024).
23. TensorFlow Lite TinyML for ESP32. Available online: <https://eloquentarduino.com/posts/tensorflow-lite-tinyml-esp32> (accessed on 30 March 2024).
24. Suresh, P.; Daniel, J.V.; Parthasarathy, V.; Aswathy, R. A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment. In Proceedings of the 2014 International Conference on Science Engineering and Management Research (ICSEMR), Chennai, India, 27–29 November 2014; pp. 1–8.
25. Tekin, N.; Acar, A.; Aris, A.; Uluagac, A.S.; Gungor, V.C. Energy consumption of on-device machine learning models for IoT intrusion detection. *Internet Things* **2023**, *21*, 100670. [[CrossRef](#)]
26. Koirala, A.; Bista, R.; Ferreira, J.C. Enhancing IoT device security through network attack data analysis using machine learning algorithms. *Future Internet* **2023**, *15*, 210. [[CrossRef](#)]
27. The TON_IoT Datasets: UNSW Research. Available online: <https://research.unsw.edu.au/projects/toniot-datasets> (accessed on 30 April 2024).
28. Elnakib, O.; Shaaban, E.; Mahmoud, M.; Emara, K. EIDM: Deep learning model for IoT intrusion detection systems. *J. Supercomput.* **2023**, *79*, 13241–13261. [[CrossRef](#)]
29. Qureshi, A.U.H.; Larijani, H.; Mtetwa, N.; Javed, A.; Ahmad, J. RNN-ABC: A new swarm optimization based technique for anomaly detection. *Computers* **2019**, *8*, 59. [[CrossRef](#)]
30. Sahu, A.K.; Sharma, S.; Tanveer, M.; Raja, R. Internet of Things attack detection using hybrid Deep Learning Model. *Comput. Commun.* **2021**, *176*, 146–154. [[CrossRef](#)]
31. Alotaibi, B.; Alotaibi, M. A stacked deep learning approach for IoT cyberattack detection. *J. Sens.* **2020**, *2020*, 8828591. [[CrossRef](#)]
32. Tomer, V.; Sharma, S. Detecting iot attacks using an ensemble machine learning model. *Future Internet* **2022**, *14*, 102. [[CrossRef](#)]
33. De Souza, C.A.; Westphall, C.B.; Machado, R.B.; Sobral, J.B.M.; dos Santos Vieira, G. Hybrid approach to intrusion detection in fog-based IoT environments. *Comput. Netw.* **2020**, *180*, 107417. [[CrossRef](#)]
34. Kumar, P.; Gupta, G.P.; Tripathi, R. A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 9555–9572. [[CrossRef](#)]
35. Kumar, R.; Kumar, P.; Tripathi, R.; Gupta, G.P.; Garg, S.; Hassan, M.M. A distributed intrusion detection system to detect DDoS attacks in blockchain-enabled IoT network. *J. Parallel Distrib. Comput.* **2022**, *164*, 55–68. [[CrossRef](#)]
36. Nasir, M.; Javed, A.R.; Tariq, M.A.; Asim, M.; Baker, T. Feature engineering and deep learning-based intrusion detection framework for securing edge IoT. *J. Supercomput.* **2022**, *78*, 8852–8866. [[CrossRef](#)]
37. Almogren, A.S. Intrusion detection in Edge-of-Things computing. *J. Parallel Distrib. Comput.* **2020**, *137*, 259–265. [[CrossRef](#)]
38. Hindy, H.; Bayne, E.; Bures, M.; Atkinson, R.; Tachtatzis, C.; Bellekens, X. Machine learning based IoT intrusion detection system: An MQTT case study (MQTT-IoT-IDS2020 dataset). In *Selected Papers from the 12th International Networking Conference: INC 2020*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 73–84.
39. Koroniots, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. [[CrossRef](#)]
40. Pearson, K.X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *London Edinburgh Dublin Philos. Mag. J. Sci.* **1900**, *50*, 157–175. [[CrossRef](#)]
41. Hotelling, H. Analysis of a complex of statistical variables into principal components. *J. Educ. Psychol.* **1933**, *24*, 417. [[CrossRef](#)]
42. Li, J.; Othman, M.S.; Chen, H.; Yusuf, L.M. Optimizing IoT intrusion detection system: Feature selection versus feature extraction in machine learning. *J. Big Data* **2024**, *11*, 36. [[CrossRef](#)]
43. Ngo, V.D.; Vuong, T.C.; Van Luong, T.; Tran, H. Machine learning-based intrusion detection: Feature selection versus feature extraction. *Clust. Comput.* **2023**, *27*, 2365–2379. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.