QATAR UNIVERSITY

COLLEGE OF ENGINEERING

SMART HARDWARE TROJAN DETECTION SYSTEM

BY

IYAD W. J. ALKHAZENDAR

A Thesis Submitted to

the College of Engineering

in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Computing

June 2022

COMMITTEE PAGE

The members of the Committee approve the Thesis of
Iyad W. J. Alkhazendar defended on 17/05/2022.

Dr. Uvais Ahmed Qidwai
Thesis/Dissertation Supervisor

Dr. Elias Yaacoub
Committee Member

Dr. Syed Rafay Hasan
Committee Member

Approved:

Khalid Kamal Naji, Dean, College of Engineering

ABSTRACT

ALKHAZENDAR, IYAD W.,Masters: June : 2022, Masters of Science in Computing

Title: Smart Hardware Trojan Detection System

Supervisor of: Prof. Uvais A. Qidwai.

The IoT has become an indispensable part of our lives at work and in our home applications. Due to the need for many IoT devices, IoT manufacturers are least concerned about security vulnerabilities during designing and developing these devices. Because of this, it becomes easier for adversaries to manipulate the hardware and insert Trojans or Remote File Inclusion to control remotely. This thesis aims to build a model to identify hardware Trojans in IoT devices using multiple machine learning models. We used different machine learning models to evaluate the performance and accuracy. In addition, we chose a distinctive feature that can detect the presence of Trojan in these devices. The proposed model is estimated using a smart city testbed and existing and real-time datasets generated. The testbed used was designed to simulate and assess the Hardware trojan attacks such as the DOS attack and covert channel attack. We could measure the power profile and network traffic on the IoT gateway device to analyze the performance and accuracy using the real-time dataset to detect the attacks.

DEDICATION

I dedicate my dissertation work to my loving wife and my beloved kids. A special feeling of gratitude to my mother, brothers, and sister for the encouragement to finish my master's and thesis.

ACKNOWLEDGMENTS

I would like to acknowledge the Support of Qatar University for providing all the needs to achieve the requirements of this study, and I would like to thank my thesis supervisor Prof. Uvais for his support, also thank Prof. Mohsen Guizani and Prof. Amr Mohamed for their guidance through my study.

Not least of all, I would like also to express my gratitude to all of the faculty members who taught during my master’s study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Internet of Things (IoT) devices are widely used in the intelligent city network due to their defined resources. Millions of devices are interconnected through the internet around the world. Smart cities integrate the infrastructure of the different domains and cyber-physical technologies to improve economic efficiency and the quality of life by reducing power waste and intelligent management of the transportation traffic [1]. Smart cities are also known as intelligent or digital cities, where extensive internet and communication (ICT) applications are used to collect the information from various Internet of Things (IoT) devices, sensors, and actuators to augment an innovation which leads to a reduction in cost and applied resources of the system. Furthermore, it strengthens the link between government and citizens. Smart cities are genuinely dependent on IoT devices, and a sizeable attacking surface characterizes that. Smart cities are supposed to help improve services for people but being irresponsible with data storage could result in privacy violations, and poorly implemented security could allow cyber attackers to interfere with services and systems people need. Security of IoT systems has been identified as one of the most challenging tasks for intelligent cities [2]. Sensors and internet-connected devices may improve urban services but could also be used by hackers and foreign states to disrupt or spy. In addition, attacks on an IoT device's embedded Integrated circuit (IC) have exponentially increased after integrating IoT devices in different domains (such as healthcare, transportation, Radar and CCTV surveillance, etc.). As a result, multiple attacks target the IoT devices' security, including confidentiality, integrity, availability, authentication, and privacy [3]. According to the BBC report, the Attacker performs a grid-lock attack to compromise the traffic control system of the 21sˆt century city affecting people's lives and work [4]. In 2018, Atlanta was under a ransomware attack, and the Attacker targeted applications used by customers to pay their bills. As a precaution against this attack, Hartsfield-Jackson Airport shut down the Wi-Fi network, and 8,000 people unplugged their systems from the intelligent infrastructure [5]. To defend against such an attack, a traditional security mechanism cannot be deployed. As the traditional security mechanism would cause a computational cost and requires high processing power to enable such a mechanism [6]. Improved technology should counter the new threats and ensure the IoT system's security [7]. Around the world, security specialists and researchers pay attention to the smart cities sector to strengthen the security of the IoT system, proposing various frameworks and Hardware Trojans Detection System (HTDS) mechanisms to avoid such types of cyber- attacks. Hence, there is a requirement to study and analyze using predictive Machine Learning (ML) to detect and block the malicious patterns of the IoT network. Intelligent cities generate enormous amounts of data from IoT devices, challenging the HTDS. This massive computational storage of uncorrelated and redundant data adversely affects the detection mechanism, resulting in high false positives and high false negatives. The assessment of the IoT system vulnerabilities is perspective using pyramid-of-pain (PoP) [8].

Figure. 1 describes the PoP, and it has six different levels [9]. The topmost layer consists of sensors and actuators; this layer provides a potential attacking surface due to its heterogeneous environment. The communication layer is an intermediate between the sensor and Data collection layer. This layer is prone to attacks when data is in motion and transmitting, where the IoT device integrity and data confidentiality breaches if the communication layer gets compromised. The hardware abstraction layer facilities the interaction between the application programming. The Firmware

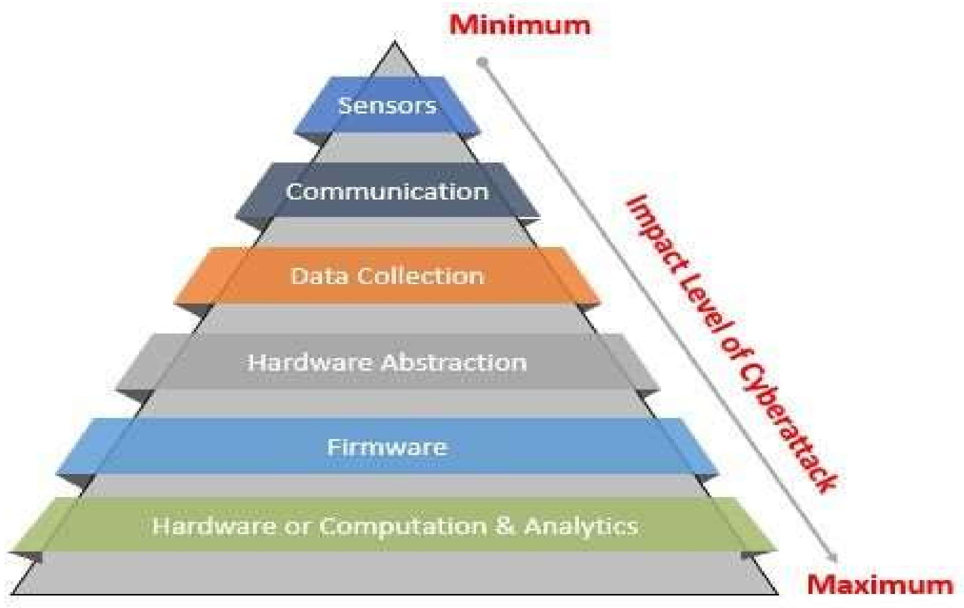


Figure 1. IoT vulnerabilities Pyramid-of-Pain [8]

layer is the interface (API) with the sensor data. The final layer comprises different ARM-based IC such as field-programmable gate arrays (FPGAs) and System on Chip (SoC). An Intellectual property (IP) core behaves like a third party to deliver the computational processing in the case of FPGAs/SoCs. The pyramid encapsulates the vulnerabilities of the IoT environment; the consequence of an attack damaging the IoT system that increases from top to bottom. The attack's impact on the last layer is maximum (i.,e Hardware or computational and analytics layer), which this thesis investigates comprehensively.

Thesis Objectives and Contributions

We proposed a novel Hardware Trojan Detection System (HTDS) mechanism in this research. So, the system can continuously monitor an IoT device's network data and power consumption by merging the IoT devices' network traffic and power profile data and analyzing and detecting the Trojan in the IoT environment based on the

Machine Learning (ML) trained models. Furthermore, using Artificial Intelligence (AI) makes the defense system adaptable to newer attacks or attacks with minor/significant previous versions. We analyze the models with publicly available datasets and our generated datasets used for testing the HTDS. The contribution of this thesis is as follows:

- Build a testbed to simulate two of the most critical HT attacks on the IoT Edge devices (DOS attacks and Covert channel attacks).
- Generate and collect a dataset of power profiles for the IoT devices in Normal and attack cases to train the model.
- Builds a real-time supervised lightweight multiclass machine learning model using different machine learning models and shows which is the best among them for detecting DOS and Covert channel attack using features from the network traffic packets and power profile of the IoT Device.
- Proof that the proposed machine learning model can detect DOS attacks and CC attacks in all possible attack scenarios including concurrent attacks.

Thesis Overview

The flow of the thesis is organized as follows:

- Chapter 2: Background and the related work of existing HTDS.
- Chapter 3: Explain the Hardware Trojan detection System model and describe the proposed model's methodology.
- Chapter 4: Results and the Evaluation discussion.
- Chapter 5: Conclusion and future direction.

CHAPTER 2: BACKGROUND AND RELATED WORK

2.1 Background

2.1.1 IoT Devices categories

IoT devices can be categorized into two categories. First, IoT – Edge devices (IoT-ED) usually have low computational power, low bandwidth, and minimal power consumption. It is mainly for connection to sensors for reading various information, and usually, it is low constrained, the second category is the IoT – Gateway devices (IoT-GW), which have more resources than the IOT-ED, which receives all the network traffic packets and power consumption of the IoT-ED and analyzes the package to indicate if the IoT-ED is under attack.

2.1.2 Network Traffic (NT)

All the IoT devices that are connected to the IoT-GW send the network traffic, which consists of data transmission to reception ratio, duration of the activity, transmission mode, source IP address, destination IP address and data value information, etc., the network traffic packets are different depending on the status of the communication, for that when there is an attack established or being triggered because of the HT, the number of packets and other packets fields is different according to the type of attack.

2.1.2 Power Profile (PP)

The power consumption data, voltage, and current of the IoT-ED are logged on the IoT GW to be analyzed. A sensor INA260 is an example of a sensor that can read the current, voltage, and power consumption that will be used as part of the testbed to get a reading of the power consumption of IoT-ED.

2.2 Related Work

2.2.1 Hardware Trojan Detection Mechanism

The attacks on the IoT hardware are considered one of the most scathing attacks on the system, Hardware Trojans that could be implanted on the Integrated circuits (ICs) during the designing and manufacturing stages. These Trojans on the system compromise the IoT device or system by performing many attacks such as Denial of Service (DoS), data leakage, and covert channel attacks. Moreover, Trojans could change the functionality of the IC and provide backdoor services to access the hardware or firmware of the IoT devices to manipulate the data [10]. Many researchers have found high- security risks in the hardware of IoT devices. On a single chip, various integrated gates accommodate such chips labeled as extensive integration in the Very-large-scale-integration (VLSI) circuits, The Ultra scale integration (ULSI), the System on Chip (SoC), and the Network on Chip (NoC). For an adversary to implant Trojans or malware on the hardware level of the device during production [11]. These risks indicate the urgency of how IC security is essential, besides it has a few challenges, namely: (1) It is challenging to distinguish because of the small scale between the noise and the Hardware Trojans (HT) [12]. (2) Activation time is always connected to a rare event, making it difficult to predict when HT is activated [13]. (3) The continuous expansion of chip platforms became highly complicated, increasing and diversifying the number of attacks [14]. Due to these challenges, researchers showcase their interest in preventing HT by designing an HT detection mechanism from the Attacker's perspective. The authors [15] proposed a feature matching methodology to capture the trojan features. In last they abstract these features to mitigate the Trojan circuitry considering the following categories AES-T100, AES-T600, AES-T700, AES-T1600, Basic RSA-T100, Basic RSA-T300, UART, and

OR1200 CTRL; these Trojans used the unspecified pins for leaking information without giving any indication to the monitoring side-channel signals. In [16], a Trojan detection mechanism has been proposed. They use an infested architecture based on the Zynq-7000 programmable SoC bus platform. The study showed that the communication between them is the Trojan Channels, and the bus topology operated on the idle bus cycle. Authors in [17] propose a stable mapping method," dynamic-resource-management dependent on security value (DRMaSV)." In this research, a standard mapping method called dynamic resource management based on security value (DRMaSV) to enhance Coarse-grained re-configurable-architectures (CGRA) operation toward secure hardware mapping. The methodology is to attach the Dynamic Resource Management approach to standard mapping CGRA, and this approach shows the effectiveness of the proposed secure mapping method for SoC and simulation. Mohammed et al. [18] implement a technique of detection based on utilizing the power profile (PP) and the network traffic (NT) data without- intervening with IC design for the detection of any malicious activity; this approach can detect multiple attacks such as DoS, ARQ, power depletion, covert channel attack, and impersonation attack; also the authors implemented data fusion to combine both the PP and NT with accuracy for each one separately around 99%, and if it's all in concurrent the accuracy increased after the researcher did the data fusion approach, which considering as one of the states of the art on the detection of HT that achieving high accuracy and can detect concurrent attacks without any time intervention. Many authors use fusion techniques. The fusion concept mainly comprises three levels based on the processing fusion stages. On every level, the representation of the information is different from one another.

Table. 1. Existing HDTS approaches

Article	Selected Features	No. of samples (Training & Testing)	Classifier	Fusion Technique	Evaluation metrics
[23]	Power Utilization	360k& 120K	BPNN	Wavelet translation	sensitivity =99.2%
[24]	Propagation delay	271 & 1329	K-NN, DT, BC	PCA/FLD	Accuracy = up to 95%
[25]	Transient Supply current	N/A & >560	Cluster ensemble	UCFS	Accuracy = up to 93.57%
[26]	EM Traces	150 & 50	one-class SVM	PCA	Accuracy = up to 99.4%
[27]	Transmission Power	30 & 90	one-class SVM	PCA	FPR=FNR = 0%
[28]	Ordered Mixed Features GEP (OMF-GEP)	-	-	GEP algorithm	Fitness rate = above 90%
[18]	Power profile and Network data	8,432,640	Random Forest	-	Accuracy = 99%
[29]	Functional & Structural Features	800 & 11	Voting Ensemble	-	TPR = 100 % TBR = 98%
[30]	Single Channel Images	Trust-HUB	CNN	HERO	Accuracy = 97.51%
[31]	Testability analysis & Reinforcemen t learning	NA & 7838 Trigger coverage signal	Reinforce ment learning	-	Accuracy = average of 96%

The data and feature fusion are states, characteristics, and features, whereas the output information of the decision level is the decision [19]. Various data fusion techniques are applied at different levels to perform better data processing. The following are the fusion levels of data fusion: Low-Level Fusion (Data Level Fusion) in this level, the raw data from the Different sources are aggregated to generate informative data [20]. Feature Fusion Level (Mid-Level Fusion) Different data features are combined to extract optimal features by applying feature selection and dimensionality reduction techniques [21]. Decision fusion (High-Level Fusion) The decision of different models is fused to make a final decision. At this level, the approach of each model carried out the operation of detection, data exploration, and dimensionality reduction on the observed patterns of the network [22]. Table 1. describes the existing approaches for Hardware Trojan detection. Finally, all these models are fused into a comprehensive model to make an accurate decision based on this level. The two main categories of feature fusion for the HTDS system are filter and wrapper methods. In filter method approach, it follows a statistical approach based on information theory, namely Canonical Correlation Analysis (CCA) [32], Principal Component Analysis (PCA) [33], Latent Dirichlet Allocation (LDA) [34], Correlation-based Feature Selection (CFS) [35] etc. Various fusion techniques are used, highlighting the classical approaches of feature fusion techniques. Other Techniques used by the authors depend on the power profile of the trusted devices and, according to the trained model of the dataset from the trusted device, can detect the attack on IoT Device.

2.2 Threat Model and Attack scenarios

2.2.1 Threat Model

IC chips are one of the main components of all IoT Edge devices. In This threat model, the Attacker during the IoT Edge devices manufacturing could implant HT and change the IC chip's internal structure. Furthermore, the Attacker could program the HT to make an attack whenever the trigger is initiated; such critical attack that can be triggered is the DOS attack and covert channel attack.

2.2.2 Attacks Scenarios

- Under the Normal Case, the IoT-ED will send both the Network packets and the power consumption usually to the IoT-GW as per the actual packets sent. The power consumption for the regular activity used by IoT-ED will usually be a timer to send the data to the IoT-Gateway device.
- In case of a DOS attack, which tries to get the IoT-ED to be unavailable due to the floods on the targeted device with repeated messages, or even to attack the IoT-GW. Network packets and power consumption of the IoT-ED will be sent to the IoT-GW during the attack.
- Another Attack scenario is that Network covert channels are an evasion technique that employs available protocols to transmit unauthorized information; covert channels are divided into two main types of storage and timing. Our scenario will test the Storage covert channel attack, specifically TCP covert channel.

After simulating these types of attacks scenarios, we found that our approach and techniques used to detect the attacks can detect all possible scenarios, even in case the attack was targeting another IoT-ED and the network traffic packets were not sniffed by the IoT-GW, the model in IoT-GW was able to detect the attack using the power profile alone, also another scenario when both attacked initiated at the same time, the machine learning model was able to detect concurrent attack with a high accuracy, which proves that our approach and technique using network traffic and power profile would detect all possible scenarios with high accuracy.

CHAPTER 3: SYSTEM MODEL AND METHODOLOGY

3.1 System Model

The proposed scheme enhances the security of an intelligent environment where various protocols are used to transfer the data from devices to the cloud. The data is very confidential; in some situations, The Hardware trojan tries to disrupt the communication of a device itself or steal some critical data, leading to the breach of personal data. An appropriate solution detection mechanism is adopted to avoid such attacks in an intelligent network. Hence, various ML/DL models detect the negative pattern in the network. In the usual scenarios, all live traffic packets of the IoT edge devices are monitored by the cloud when enabling a detection mechanism against attacks on the cloud to analyze the packets but due to resource constraints of IoT devices, computational cost, and high processing power it is not optimum to do the detection, for that HTDS smartly classifies the attack patterns of typical anomalies from IoT edge devices directly and alerts the admin once the IoT device/system is under attack. A novel HTDS for the IoT devices for smart cities is depicted in figure 2.

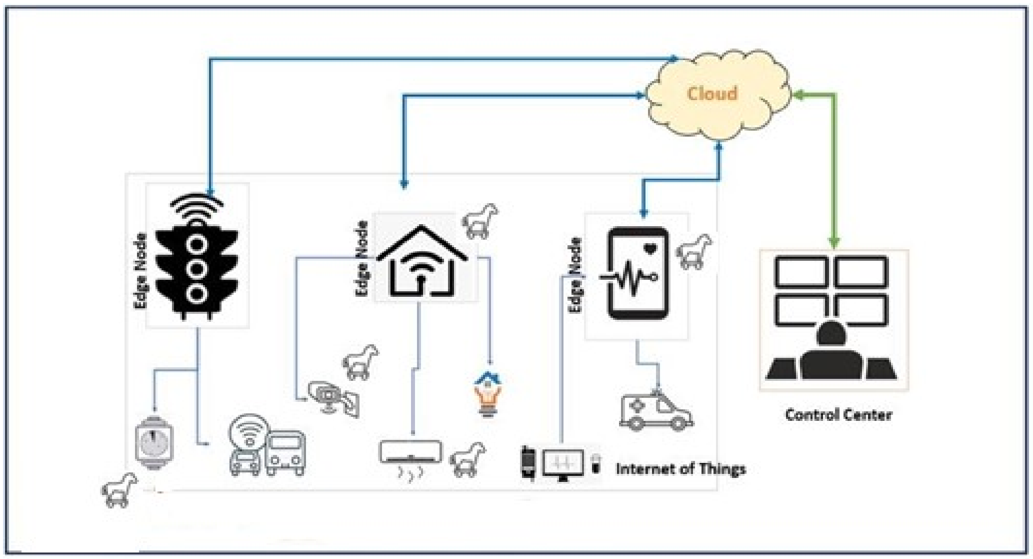


Figure 2. System Model

3.2 Methodology

The following three steps are needed to propose the desired system. The steps can be called levels or phases to develop and integrate to build the final HT-detection system (HTDS), as shown in Figure 3.

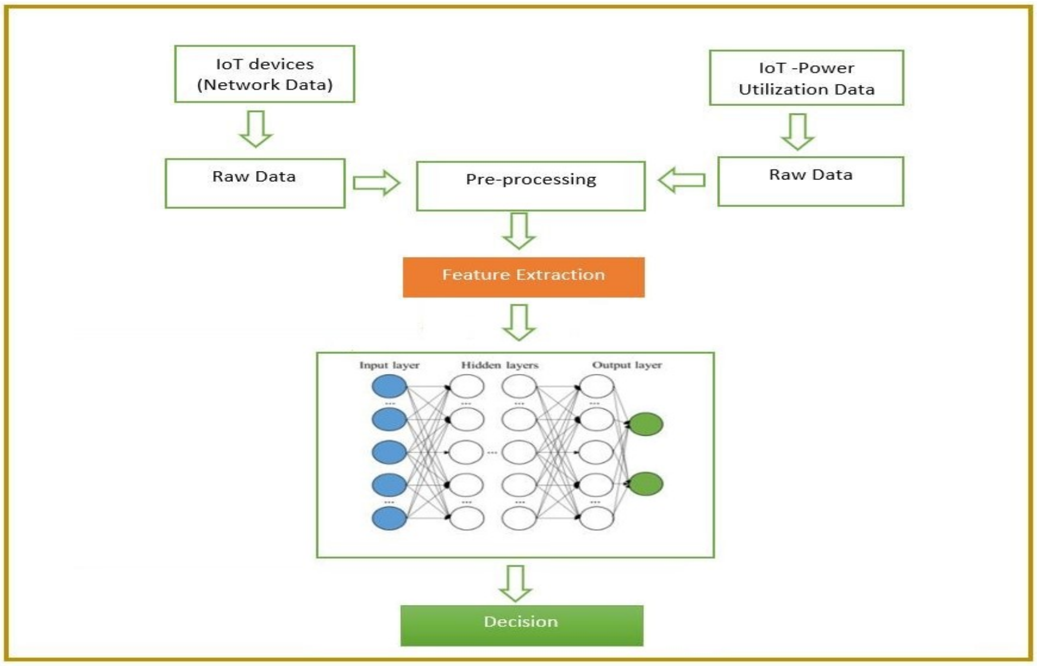


Figure 3. HT-detection model

3.2.1 Data generation phase:

This level consists of many IoT devices of various domains in a real-time scenario. Meta-information is generated from this phase in the form of raw data from multiple domains in the smart city environment such as transportation, smart home, and e-health system. These domains work on both wireless/wired protocols to transfer confidential information from IoT devices to the edge node of the particular field. We utilized the IoT device's network data and power utilization. The network traffic is being captured or sniffed using a sniffing module, i.e., (the Wireshark tool), and another IoT device monitors the Power utilization data of the IoT under attack. We assumed that if the Attacker tries to compromise the IoT-1 device by sending Remote File Inclusion (RFI) or Trojan payloads, we observe the negative patterns in the network traffic. These payloads will deviate from the IoT device's voltage and current under attack parameters. In such a case, we will fail to detect the Trojans on the IoT devices because the detection model will not receive voltage and current parameters from that device under attack. Hence, our model uses both (Network + Power utilization) data to detect the malicious activities that result in the loss/death of IoT devices from the network. For example, if the IoT is in the healthcare domain, it threatens the patient's life. To avoid abrupt circumstances, we tune the power utilization parameters of the IoT devices such that every IoT device should share the voltage and current parameters with the other IoT devices. The information flow of the network can be obtained from the traffic bit with the header section. The data collected here is raw data, which consists of basic information about the packets. More details (features) are extracted in the next level.

3.2.2. Feature Extracting

The feature extracting will be from PCAP using Wireshark (tshark tool) and data cleaning The network packets after its generated by using the covert_tcp.c tool for the covert channel and the packets that been captured during the DOS attack in PCAP format, data needs to be changed to CSV file, the tshark can extract the ID, checksum values of all datagrams as in the following an example command :

```
tshark -r input.pcapng -T fields -E separator=, -E header=y, -e ip.id -e ip.checksum >
output.csv
```

then processing of the data is required to delete any unnecessary blank rows, and then a label needs to be added to the data if it's normal. DOS, or covert attack.

3.2.3 Feature Selection

Feature selection is crucial for the result to be accurate, Feature engineering is used for both DOS and Covert channel (TCP sequence) the following feature will be used: TCP Sequence, TCP next sequence, TCP acknowledgment, TCP Checksum, TCP SYN flag, and for the power the selection will be for the current, voltage and the power consumption.

3.2.4 Feature Engineering

Before processing the training data, a Feature with hexadecimal content like the TCP checksum and IP ID needs to be converted to decimal to be processed. Also, all the rows that have blank values need to be deleted before it's passed to the machine learning model.

3.2.5 Feature Scaling

All features should be scaled before we implement the training on the ML model, as it ensures the values are in the same range and reduces any error. We will use for the scale of features the StandardScaler from sci-kit.

3.2.6 Feature merging phase:

In this phase, the network Power utilization of the data is preprocessed. And The network data consist of data transmission to reception ratio, duration of the activity, transmission mode, source IP address, destination IP address and data value information, etc., and power consumption data voltage and current. Other features may be added later if required during the implementation of the detection mechanism. At the same time, the Power utilization of data consists of power utilized by the IoT when the IoT device is under attack and without attack, we write a python script to combine both sniffed traffic packets and power utilization readings into one CSV file to be passed to the machine learning model after doing the feature processing and feature selection to check if the IoT device is under attack.

The objective of the feature merging technique is to make the dimensionality reduction of multiple datasets, such that we get higher accuracy of the HTDS model. The best model as per our criteria is the one with higher accuracy, precision, and Recall, To evaluate the quality of the feature. We are concerned about the training and testing time of the classifier to be minimum; usually, the time required to train the classifier is more than the testing time, which affects the performance of the model.

3.2.7 Model training:

The dataset is split into training and test data sets to be used by the ML model. 70% of the dataset is for training, and 30% is for testing, We will use the method under the sci-kit learn library to split the data set to training and testing 'train_test_split.'

3.2.8 Model Evaluation

The evaluation of the trained model will be done on different techniques such as the accuracy score, cross-validation, precision score, recall score, and F1 score and more clarification will be discussed in section 4.3.

3.3 Decision Phase:

In this detection phase, the merged features input the ML model to predict the network/power consumption activity as normal or malicious. Initially, a model is formed from which the features are fed to build a training framework categorizing malicious and benign behavior: ML is employed to predict the network and power consumption activity. Then, the network / Power consumption activity features are compared with those of the training phase. The HTDS is designed to manage a wide range of attacks. In the detection phase, the features of the actual data packets are extracted from the collected raw data. Later the elements are classified by comparing with the training sequence, finally, the decision is made as regular network activity, DOS attack, or CC attack.

3.4 Testbed Setup

The innovative environment testbed consists of the following devices with the following specification:

IoT–Gateway : Raspberry Pi specification: SoC: Broadcom BCM2711B0 quad-core A72 (ARMv8-A) 64-bit @ 1.5GHz, GPU: Broadcom VideoCore VI, Networking: 2.4 GHz and 5 GHz 802.11b/g/n/ac wireless LAN, RAM: 4GB SDRAM, Bluetooth: Bluetooth 5.0, Bluetooth Low Energy (BLE), GPIO: 40-pin GPIO header, populated , Storage: microSD, Ports: 2 × micro-HDMI 2.0, 3.5 mm analogue audio-video jack, 2 × USB 2.0, 2 × USB 3.0, Gigabit Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI), Dimensions: 88 mm × 58 mm × 19.5 mm, 46 g.

IOT Edge devices: Raspberry Pi 3 with following specification : Clock frequency: 1.2 GHz ,Chipset (SoC): Broadcom BCM2837, Processor: 64-bit quad-core ARM Cortex-A53 , Graphics processor: Broadcom Dual Core VideoCore IV (OpenGL ES 2.0, H.264 Full HD @ 30 fps), Memory (SDRAM): 1 GB LPDDR2, Number of USB 2.0 ports: 4, Port extension: 40-pin GPIO, Video outputs: HDMI and RCA, plus 1 CSI camera connector, Audio outputs: 3.5 mm stereo jack or HDMI, Data storage: MicroSD card, Network connection: 10/100 Ethernet, 802.11n Wi-Fi and Bluetooth 4.1 (BLE - Low Energy), Peripherals: 17 x GPIO ,Supply: 5V 2.5A via micro USB, Dimensions: 85.60 mm × 53.98 mm × 17 mm, Weight: 45 g.

Another IoT-Edge Device used is ESP32cam with the following specification: Ultra-small 802.11b/g/n Wi-Fi + BT/BLE SoC module, Low-power dual-core 32-bit CPU for application processors, Up to 240MHz, up to 600 DMIPS, Built-in 520 KB SRAM, external 4M PSRAM, Supports interfaces such as UART/SPI/I2C/PWM/ADC/DAC, Support OV2640 and OV7670 cameras with built-in flash, Support for images Wi-Fi upload, Support TF card, Support multiple sleep modes, Embedded Lwip and FreeRTOS, Support STA/AP/STA+AP working mode, Support Smart Config/ AirKiss One-click distribution network, Support for local serial upgrade and remote firmware upgrade (FOTA), Support secondary development. ESP32-cam as IoT-Edge node is connected with sensors such as surveillance cameras, temperature sensors, etc. ESP32 devices sense the data of the various sensors and transfer them to the Gateway device of the IoT system. Figure4 describes the testbed scenario, The Raspberry-PI4 acts as a Gateway node, and it receives data from the ESP32 through Wi-Fi. The Gateway device preprocesses the data and transfers it to the cloud for future records, analytics, and visualization.

INA260 Sensor: Power Profile will be sent to the IoT-GW through which will be connected from IoT-ED to the IoT-GW the sensor has the following specification: Precision Integrated Shunt Resistor: Current Sense Resistance: 2mΩ, Tolerance Equivalent to 0.1%,15-A Continuous From –40°C to +85°C, ten ppm/°C Temperature Coefficient,(0°C to +125°C), Senses Bus Voltages From 0 V to 36 V, High-Side or Low-Side Sensing, Reports Current, Voltage, and Power, High Accuracy: 0.15% System Gain Error (Maximum), 5-mA Offset (Maximum), Configurable Averaging Options,16 Programmable Addresses, Operates From a 2.7-V to 5.5-V Power Supply, 16-Pin, TSSOP Package.

LCD screen: to show continuous current, voltage, and power consumption changes, Testbed devices are shown in figure 4 for more clarification.

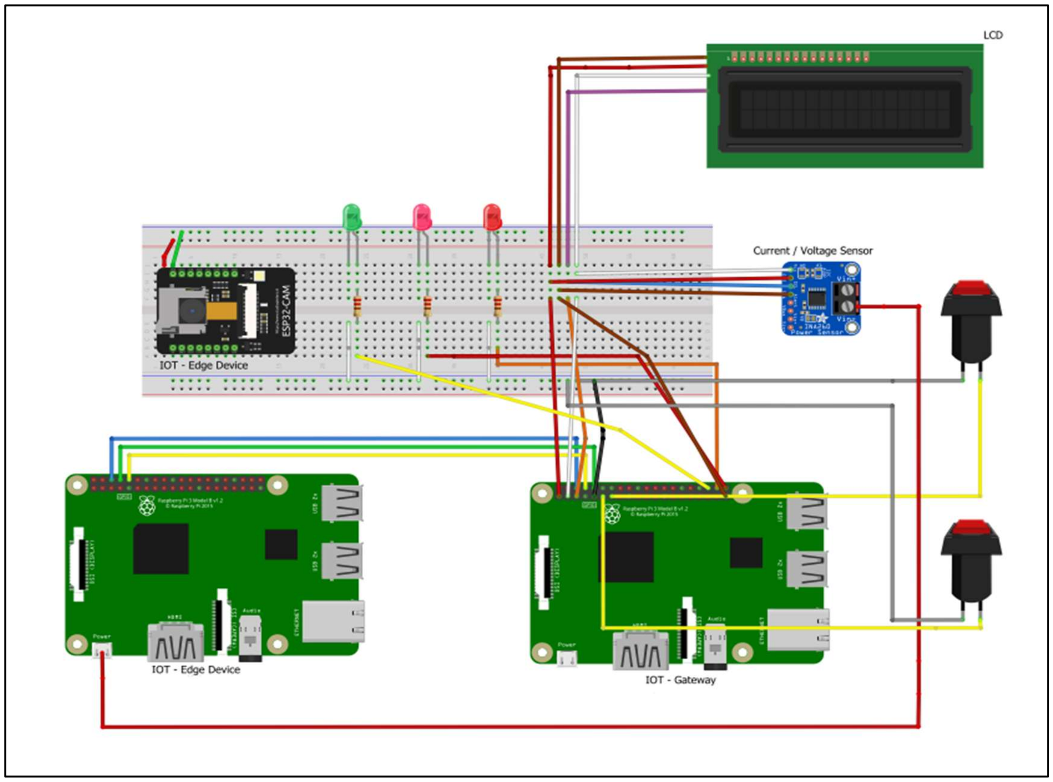


Figure 4. Experimental Setup

3.4.1 Tools used in our testbed experiment:

Wireshark and Scapy to sniff the packet data from the IoT-Edge Devices.

INA260 sensor for reading and sending the power profile to the IoT-GW device.

Covert_tcp tool for generating the covert channel attack packets.

3.4.2 Attack Scenarios Explanation:

Scenario 1: IoT device1-Edge device sends a regular traffic and power consumption to the IoT-Gateway device. While the IoT device2-Edge device (ESP32-CAM) streams and sends traffic data to the IoT-Gateway device, capturing traffic packets through the Wireshark tool and reading the power profile data through the INA260 sensor. And Both reading is saved combined on a CSV file on the IoT -Gateway.

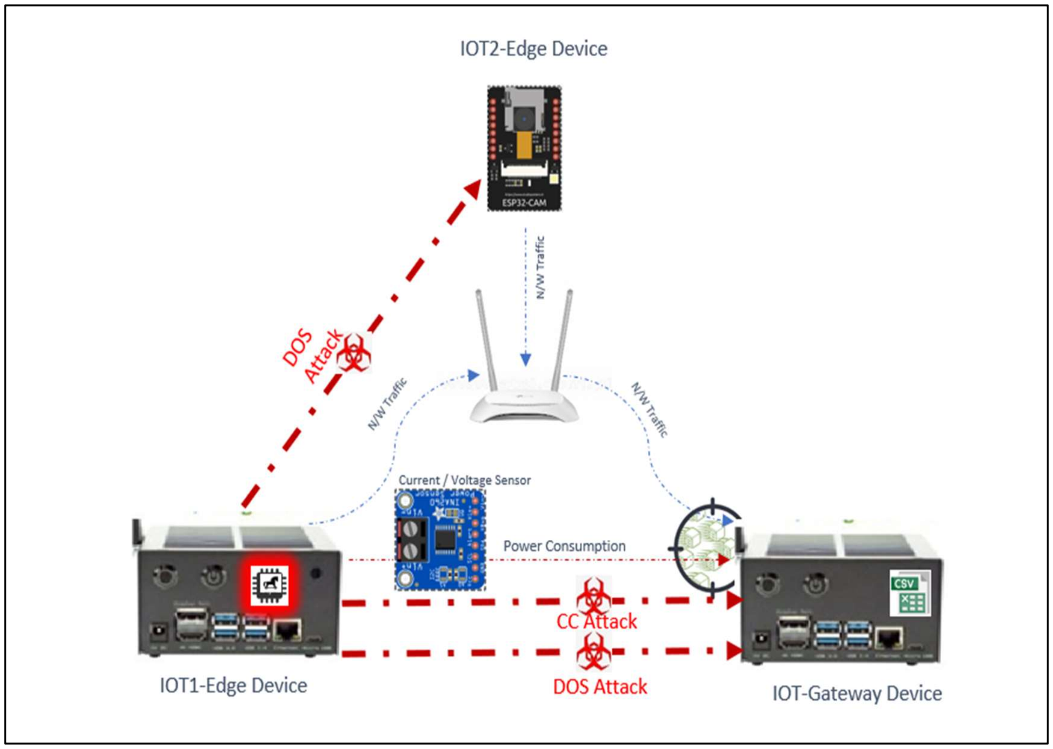
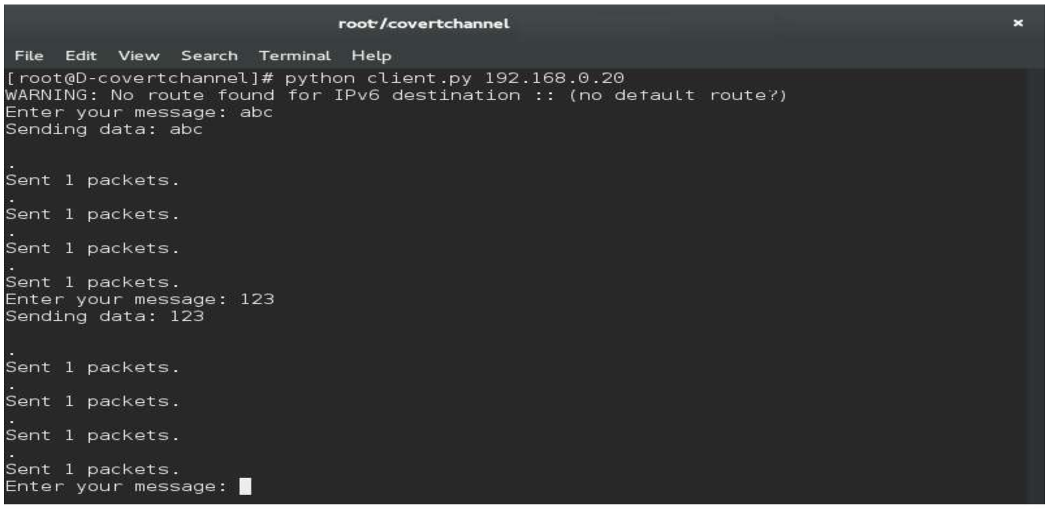


Figure 5. Attack scenarios

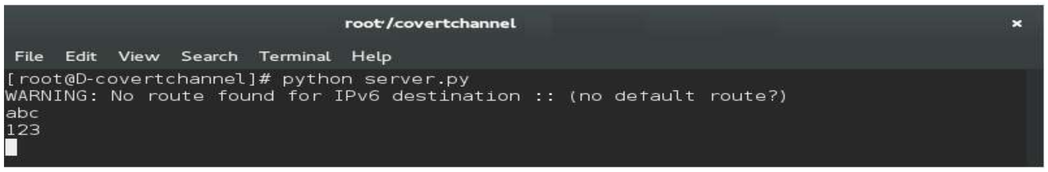
Scenario 2:In our setup, we have performed a Denial of Service (DoS) attack by sending a malicious (Trojan) payload from IoT device1-Edge device to IoT device2-Edge device, which is triggered by HT (for demonstration purposes Triggering is done by push button). The DOS attack will disrupt the communication between the ESP32-cam (IoT device2-Edge device) and the Gateway device (IoT-Gateway device) with a flood of malicious traffic, the IOT1-Edge Device will keep sending the Network data, and power consumption during the attack as shown in figure 5, both sniffed network traffic packets and the power consumption will be merged to one dataset using a python script to save it to one CSV file to be used as an input dataset for the ML model, and ML model will detect the attack.

Scenario 3: In our setup the for demonstration purpose of the Covert channel attack, the attack will be triggered by push-button from the IoT device1-Edge device and attack

IoT-Gateway the message will be sent through the IP header to the IoT -Gateway device, Scenario 4: in this scenario, we initiate the two attacks at once to see the performance of the detection mechanism in detecting concurrent attacks of the DOS and CC attacks. In our exterminate there are two programs in python, the first program is Server located on the IoT-GW device, and the second one is a Client located on the IoT-ED device, Scapy needs to be used on both Edge and Gateway devices, and the Wireshark tool must be installed on the IoT-GW to sniff the traffic packet as shown on Figure 6., both datasets of power and traffic will be combined using python script in the IoT-GW, to be passed to the machine learning model after its being processed to detect the attack once it is being initiated.

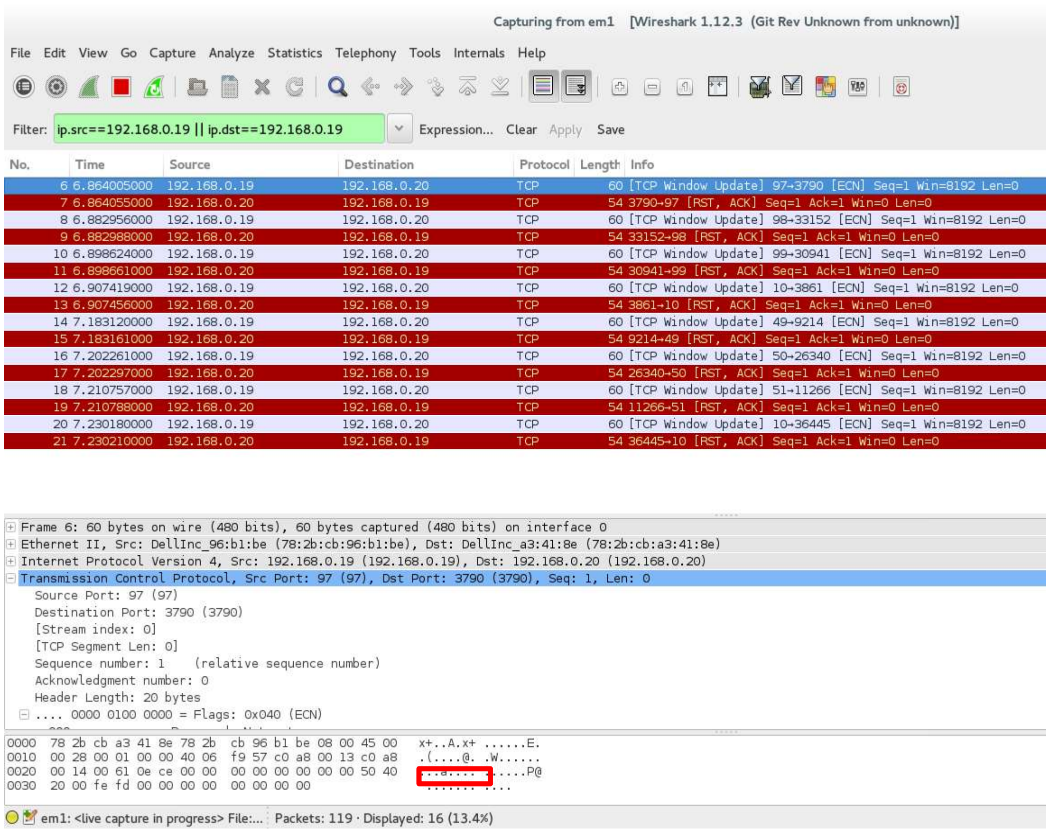


```
[root@D-covertchannel]# python client.py 192.168.0.20
WARNING: No route found for IPv6 destination :: (no default route?)
Enter your message: abc
Sending data: abc

.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
Enter your message: 123
Sending data: 123

.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
Enter your message: 
```



```
[root@D-covertchannel]# python server.py
WARNING: No route found for IPv6 destination :: (no default route?)
abc
123
```

Figure 6a. attack covert channel scenario


Capturing from em1 [Wireshark 1.12.3 (Git Rev Unknown from unknown)]

Filter: ip.src==192.168.0.19 || ip.dst==192.168.0.19

```
No.  Time         Source        Destination   Protocol Length Info
   6 6.864005000  192.168.0.19  192.168.0.20  TCP      60 [TCP Window Update] 97→3790 [ECN] Seq=1 Win=8192 Len=0
   7 6.864055000  192.168.0.20  192.168.0.19  TCP      54 3790→97 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   8 6.882956000  192.168.0.19  192.168.0.20  TCP      60 [TCP Window Update] 98→33152 [ECN] Seq=1 Win=8192 Len=0
   9 6.882988000  192.168.0.20  192.168.0.19  TCP      54 33152→98 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  10 6.898624000  192.168.0.19  192.168.0.20  TCP      60 [TCP Window Update] 99→30941 [ECN] Seq=1 Win=8192 Len=0
  11 6.898661000  192.168.0.20  192.168.0.19  TCP      54 30941→99 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  12 6.907419000  192.168.0.19  192.168.0.20  TCP      60 [TCP Window Update] 10→3861 [ECN] Seq=1 Win=8192 Len=0
  13 6.907456000  192.168.0.20  192.168.0.19  TCP      54 3861→10 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  14 7.183120000  192.168.0.19  192.168.0.20  TCP      60 [TCP Window Update] 49→9214 [ECN] Seq=1 Win=8192 Len=0
  15 7.183161000  192.168.0.20  192.168.0.19  TCP      54 9214→49 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  16 7.202261000  192.168.0.19  192.168.0.20  TCP      60 [TCP Window Update] 50→26340 [ECN] Seq=1 Win=8192 Len=0
  17 7.202297000  192.168.0.20  192.168.0.19  TCP      54 26340→50 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  18 7.210757000  192.168.0.19  192.168.0.20  TCP      60 [TCP Window Update] 51→11266 [ECN] Seq=1 Win=8192 Len=0
  19 7.210788000  192.168.0.20  192.168.0.19  TCP      54 11266→51 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  20 7.230180000  192.168.0.19  192.168.0.20  TCP      60 [TCP Window Update] 10→36445 [ECN] Seq=1 Win=8192 Len=0
  21 7.230210000  192.168.0.20  192.168.0.19  TCP      54 36445→10 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```

```
⊞ Frame 6: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: DellInc_96:b1:be (78:2b:cb:96:b1:be), Dst: DellInc_a3:41:8e (78:2b:cb:a3:41:8e)
⊞ Internet Protocol Version 4, Src: 192.168.0.19 (192.168.0.19), Dst: 192.168.0.20 (192.168.0.20)
⊟ Transmission Control Protocol, Src Port: 97 (97), Dst Port: 3790 (3790), Seq: 1, Len: 0
    Source Port: 97 (97)
    Destination Port: 3790 (3790)
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 0
    Header Length: 20 bytes
  ⊟ .... 0000 0100 0000 = Flags: 0x040 (ECN)
```

```
0000  78 2b cb a3 41 8e 78 2b  cb 96 b1 be 08 00 45 00   x+..A.x+ ......E.
0010  00 28 00 01 00 00 40 06  f9 57 c0 a8 00 13 c0 a8   .(....@. .W......
0020  00 14 00 61 0e ce 00 00  00 00 00 00 00 00 50 40   ...a.... ......P@
0030  20 00 fe fd 00 00 00 00  00 00 00 00                ........ ....
```

em1: <live capture in progress> File:... Packets: 119 · Displayed: 16 (13.4%)

Figure 6b. attack covert channel scenario

CHAPTER 4: RESULT AND EVALUATION

In this section, we used an ML model to experiment and validate the model to be deployed on the edge device of the intelligent network. ML has been chosen for fraud detection, text classification, and image recognition problems. Due to the significant outcome of ML algorithms for different situations, many researchers engage these ML algorithms to identify the abnormal behavior of the node in the network to enhance security. Machine learning models are deployed and utilized in various applications to determine the Trojans or malicious behavior in the network's traffic. We have tested ML algorithms such as Naive Bayes, K- Nearest Neighbor, Logistic Regression, Decision Trees, Support Vector Machine (SVM), Random Forest (RF), and Deep Neural Network (DNN) to choose which is the best accuracy among of them for our model. We have selected the RF algorithm among all the algorithms due to its best performance and suitability to our resource constraint architecture.

4.1 Machine learning models:

Naive Bayes: In the sub-domain (i.e., machine learning) of the AI, the Naive Bayes classifier is a simple approach that is truly dependent on the Bayes' theorem with the independent assumption among the different feature samples. It computes the probability of various classes for given input instances. Suppose X acts as a vector of wrong in-stances, where n features of an illustration are represented as x1, x2, x3, ..., xn. Considering that, there are k labels for the class (i.e., C1, C2, C3, ..., Cn). The probability condition of the Naive Bayes is P (C_k |X) Ck indicates the instance of the given X. By applying the Bayes theorem to the situation of the naive, it is represented as $P\ (C_k|x_1, x_2, x_3, ..., x_n) = \frac{P(x1,x_2,x_3,...,x_n| C\ k).P(Ck)}{P(x1,x_2,x_3,...,x_n)}$ (1)

$P(C_k|X) = \frac{P(X|C_k).P(C_k)}{P(X)}$ (2)

, According to the naive condition independent assumption, each feature is independent of the other elements. Hence, it can be rewritten as

$P(C_k|X)) = \frac{P(X|C_k).P(C_k)}{P(X)}$ (3)

K- Nearest Neighbor: The basic rule of this classification model is to classify an instance based on the distance from the k-nearest neighbor points. The final decision will be based on most of the neighbors' votes. The Euclidean distance Eq. 4 is used to measure each instance with its neighbor• Euclidean distance:

$d = \sqrt{\sum_{i=1}^{n}(X - Y)^2}$ (4)

Logistic Regression: The algorithm is well known for statistical supervised ML models. This algorithm is used for binary classification problems; this model compares the result with other ML models.

Decision Trees: The classification problem is solved by creating the tree. The tree starts at the root and stratifies the data at attributes, resulting in higher information gain (IG). It comprises sub-trees and nodes. In our scenario, the decision will appear on the leaves.

Support Vector Machine (SVM): SVM is used for the classification and regression problem; a hyperplane segregates the regular events from the plane.

Random Forest (RF) is an efficient algorithm used for both classification and regression tasks and used for feature selection. This ensemble model can attain the best accuracy among other classifiers. However, the RF algorithm model will be computationally expensive compared to other lightweight ML models in our approach.

A Deep Neural Network (DNN) is a supervised learning approach used for the classification problem. DNN comprises different layers of input, output, and hidden layers in the neural network. To enhance the performance of the DNN model, we have chosen Rectified Linear unit (ReLU) activation function in all the layers of the neural network except the output layer used for the model output. Table 2 describes the chosen hyperparameters for the DNN model. For the fast convergence, we preferred to use it as an Adam optimizer with a batch size of 60. and an epoch size of 1000 for training the model.

Table 2. DNN Hyperparameter Settings

Activation Function	ReLU
Epoch	1000
Optimizer	Adam
Learning rate	0.001
Batch size	60

The rate is set to be at 0.001. The training set is scaled using RobustScaler as it improves the performance of the DNN.

4.2 Dataset

- We capture live traffic and power data from the testbed and generate the dataset to train the machine learning model.
- During the collection of the data, different samples were collected for the DOS, CC, and the Normal traffic and power.
- 109500 samples were collected from capturing live traffic and power of the IoT-ED.
- From the samples collected a percentage of 70% was used for training and 30 % for testing.

dataset demonstrates that this method achieves higher accuracy than the other incremental models mentioned in the literature for DOS attack identification. The covert channel attack dataset was collected from the network packet from the covert_tcp.c tool to generate the traffic of the clandestine attack, Wireshark tool used to sniff the data and then converted to CVS format by a python program. For the Power profile, the dataset is generated from the reading power current, voltage, and power consumption from the IoT-Edge device by using the INA260 sensor, and last using a python script to merge both sniffed traffic packets and the reading of the power utilization to CSV file format all the scenarios (Normal case, DOS attack case, and during the Covert channel attack) to be processed and then passed to the machine learning model for the detection.

4.3 Samples:

Three samples were created for training the machine learning model:

1st sample containing regular traffic packets and reading of the normal activity of the power profile without any attack.

2nd sample contains the Traffic packets and the power profile during the DOS attack.

3rd sample contains the traffic packets and the power profile during the Covert channel attack.

4.4 Experimental Results

In the following section, we enlist the parameters and methods to evaluate the DF techniques to choose the best one as per the problem and the attack scenarios that are explained in Fig.5. The performance of the DF technique is directly proportional to the version of the HTDS model. To calculate the performance of IDS, various statistical measures are used. In binary classification, a set of samples are labeled benign or malicious classes. Generally, the performance of the ML/DL algorithms is evaluated by the metrics extracted from the confusion matrix. This section provides a detailed experimental analysis of the proposed model. We describe the metrics that are used for the evaluation of the model below. The parameters of the confusion matrix are explained below to evaluate the performance of the classification models.

- True Positive (TP):

Depict the number of malicious samples classified correctly as malicious

- True negative (TN):

Depict the number of benign samples classified correctly as harmless.

- False Positive (FP):

Depict the number of benign samples incorrectly classified as malicious.

- False Negative:

Depict the number of malicious samples incorrectly classified as benign.

Based on the confusion matrix, the metrics that are used to evaluate the performance of the ML models are explained as follows:

1) Accuracy: The ratio of correctly identified records to the complete test dataset. Accuracy is considered a suitable parameter for the test dataset, which contains balanced classes as shown in equation 5.

$$Accuracy = \frac{TP+TN}{TN+TP+FN+F} * 100 \qquad (5)$$

2) F1-Score: The harmonic mean of Recall and precision is the harmonic mean. It is defined as:

$$F1 - measure = \left(\frac{Recall * \text{precision}}{Recall + \text{precision}}\right) * 2 \qquad (6)$$

The accuracy and f1-score as shown in equation 6 of the proposed model are shown in Figure 7 and Table 2. The results demonstrate that most ML models performed similarly in both metrics, among which KNN, DT, and RF gave more than 98% accuracy. For IoMTs, accuracy and computational capability, and memory consumption of the ML algorithm are also significant. From the selected ones, KNN, DT, and RF are lightweight algorithms that are suitable for our scenario, but DT and

RF performance are relatively better than the KNN, while KNN has the biggest file size with approximate 130Mb, which is not suitable for the resource-constrained environment, for that we have selected RF for our model.

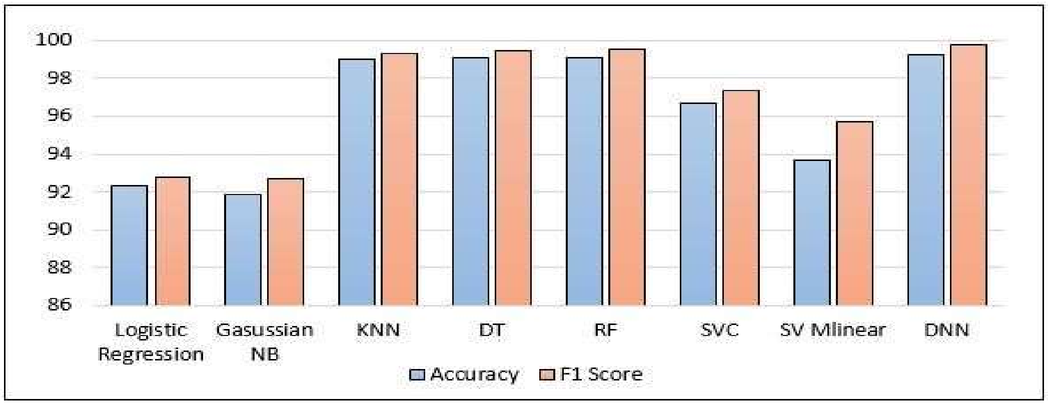


Figure 7. Accuracy and F1-Score

3) Recall measure is to understand the genuinely correct predictions.

4) Precision is the ratio of correctly recognized malicious samples to the total number of entire malicious samples.

$$Precision = \frac{TP}{TP+FP} \quad (7)$$

The precision-recall measure gives the actual performance of our model. Therefore, it is a significant metric to evaluate the classifier output quality. Like accuracy metrics, the precision, and Recall are shown in Figure. 8 and Table 3, for the selected algorithm, RF is best and very near to the highest value in the chosen list.

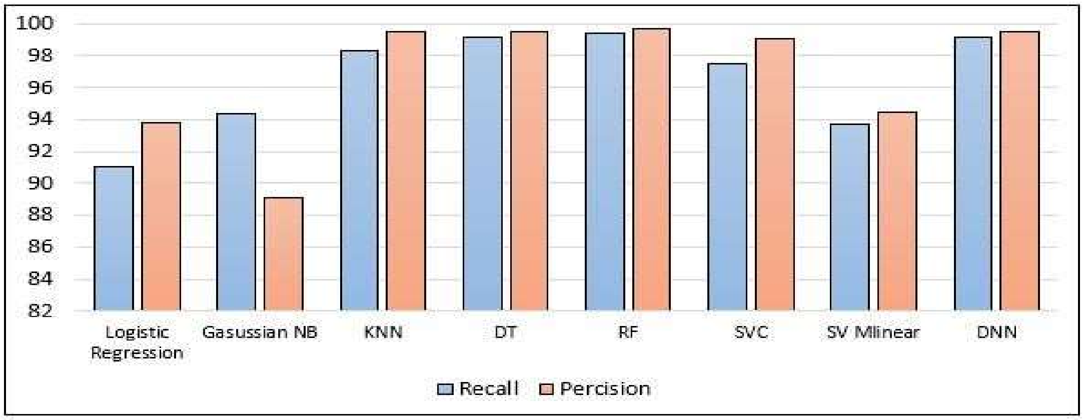


Figure 8. Recall and Precision

5) False Positive Rate (FPR): The ratio of benign samples is tagged as malicious to the complete benign samples. It is defined as

$$FPR = \frac{FP}{FP+TN} \quad (8)$$

6) True Positive Rate (TPR): It is also referred to as Recall. The ratio of correctly classified malicious samples to the entire negative samples

$$TPR = \frac{TP}{TP+FN} \quad (9)$$

In an attempt to prove the performance of the proposed model, we have included two other metrics, TPR and FPR, which correctly give the error rates of the classifier models. Figure. 9 and table 3show that the lowest TPR and FPR values are recorded for RF.

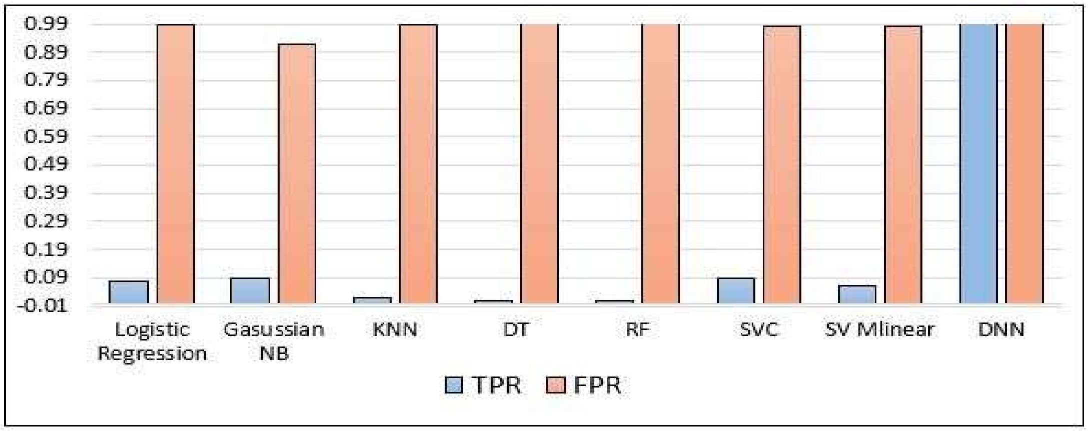


Figure 9. FPR and TPR

The Following Table 3. shows all the percentages for each Machine learning model:

Table 3. Result for each ML model

ML models	Accuracy (%)	F1-Score (%)	Recall (%)	Precision (%)	TPR	FPR
Logistic Regression	92.3	92.80	91.0	93.80	0.08	0.918
Gaussian NB	91.83	92.73	94.33	89.15	0.091	0.92
KNN	99.01	99.32	98.33	99.54	0.02	0.989
DT	99.09	99.47	99.12	99.51	0.009	0.995
RF	99.12	99.55	99.45	99.67	0.007	0.995
SVC	96.7	97.33	97.5	99.02	0.09	0.982
SVM linear	93.7	95.7	93.74	94.44	0.06	0.984
DNN	99.27	99.76	99.12	99.51	0.9912	0.9951

CHAPTER 5: CONCLUSION AND FUTURE WORKS

The Thesis reviewed the attack detection of the Hardware Trojan Detection System. And the HTDS models that followed a combination of these methods, security, and privacy of the data from various sources are significant to deal with HTDS. But our model has considered the security and privacy issues of the intelligent environment by preprocessing the steps on the IoT device itself and merging the network traffic with the data of power utilized by the IoT device. Then, this process data is transmitted to the edge node of the smart environment to the HTDS to detect the HTDS on the network. The proposed model gives us more than 99% accuracy in detecting all different scenarios including concurrent attacks or detecting any attacks based on power profile alone, which suggests the real-time detection of the proposed methodology.

Further, we would like to investigate other types of attack detection using various AI approaches.

REFERENCES

[1]. M. Mohammadi and A. Al-Fuqaha, "Enabling cognitive smart cities using big data and machine learning: Approaches and challenges," IEEE Communications Magazine, vol. 56, no. 2, pp. 94–101, 2018.

[2]. R. O. Andrade, S. G. Yoo, L. Tello-Oquendo, and I. Ortiz- Garce's, "A comprehensive study of the IoT cybersecurity in smart cities," IEEE Access, vol. 8, pp. 228922–228941, 2020.

[3]. N. B. Gaikwad, H. Ugale, A. Keskar, and N. Shivaprakash, "The internet-of-battlefield-things (iobt)-based enemy localization using soldiers' location and gunshot direction," IEEE Internet of Things Journal, vol. 7, no. 12, pp. 11725–11734, 2020.

[4]. G. Corera, "Spy bosses warn of cyber-attacks on smart cities." https://www.bbc.com/news/technology-57012725/, 2021. [On- line; accessed 29-Sep-2021].

[5]. J. Bowles, "America's cities are under cyberattack. that's bad news for IoT and smart cities." https://diginomica.com/ americas-cities-cyberattack-thats-bad-news-iot-smart-cities/,2021. [Online; accessed 29-Sep-2021].

[6]. N. A. Gunathilake, A. Al-Dubai, and W. J. Buchana, "Recent advances and trends in lightweight cryptography for iot security," in 2020 16th International Conference on Network and Service Management (CNSM), pp. 1–5, 2020.

[7]. S. Alharbi, P. Rodriguez, R. Maharaja, P. Iyer, N. Bose, and Z. Ye, "Focus: A fog computing-based security system for the Internet of Things," in 2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC), pp. 1–5,2018.

[8]. DavidJBianco, “Enterprise detection response.” http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain. html, 2013. [Online; accessed 29-Sep-2021].

[9]. V. Venugopalan and C. D. Patterson, "Surveying the hardware trojan threat landscape for the internet-of-things," Journal of Hardware and Systems Security, vol. 2, no. 2, pp. 131–141,2018.

[10]. C. Dong, Y. Xu, X. Liu, F. Zhang, G. He, and Y. Chen, "Hardware Trojans in chips: a survey for detection and prevention," Sensors, vol. 20, no. 18, p. 5165, 2020.

[11]. A.Tiwari and C. Soni, "Hardware Trojans: An austere menace ahead," in Cyber Security, pp. 349–359, Springer, 2018.

[12]. X. Chen, G. Liu, N. Xiong, Y. Su, and G. Chen, "A survey of swarm intelligence techniques in VLSI routing problems," IEEE Access, vol. 8, pp. 26266–26292, 2020.

[13]. H. Tang, G. Liu, X. Chen, and N. Xiong, "A survey on Steiner tree construction and global routing for VLSI design," IEEE Access, vol. 8, pp. 68593–68622, 2020.

[14]. W. Guo and X. Huang, "Pora: A physarum-inspired obstacle- avoiding routing algorithm for integrated circuit design," Applied Mathematical Modelling, vol. 78, pp. 268–286, 2020.

[15]. N. Q. M. Noor and S. M. Daud, "A defense mechanism against hardware trojan insertion by third-party intellectual property (ip) design blocks in aes-based secured communication system," International Journal of Information Technology, vol. 9, no. 1, pp. 87–92, 2017.

[16]. N. Fern, I. San, I. K. Koc¸, and K.-T. T. Cheng, "Hiding hardware trojan communication channels in partially specified soc bus functionality," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 9, pp. 1435–1444, 2016.

[17]. L. Liu, Z. Zhou, S. Wei, M. Zhu, S. Yin, and S. Mao, "Drmasv: Enhanced capability against hardware trojans in coarse-grained reconfigurable architectures," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 4, pp. 782–795, 2017.

[18]. H. Mohammed, S. R. Hasan, and F. Awwad, "Fusion-on- field security and privacy preservation for IoT edge devices: Concurrent defense against multiple types of hardware Trojan attacks," IEEE Access, vol. 8, pp. 36847–36862, 2020.

[19]. G. Li, Z. Yan, Y. Fu, and H. Chen, "Data fusion for network intrusion detection: a review," Security and Communication Networks, vol. 2018, 2018.

[20]. W. Ding, X. Jing, X. Yan, X. L. T. Yang, "A survey on data fusion in internet of things: Towards secure and privacy-preserving fusion," Information Fusion, vol. 51, pp. 129–144,2019.

[21]. H. Zhang, J.-L. Li, X.-M. Liu, and C. Dong, "Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection," Future Generation Computer Systems, vol. 122, pp. 130–143, 2021.

[22]. T. Omrani, A. Dallali, B. C. Rhaimi, and J. Fattahi, "Fusion of ann and SVM classifiers for network attack detection," in 2017 18th International Conference

on Sciences and Techniques of Automatic Control and Computer Engineering (STA), pp. 374–377, IEEE, 2017.

[23]. L. Ni, J. Li, S. Lin, and D. Xin, "A method of noise optimization for hardware trojans detection based on bp neural network," in 2016 2nd IEEE International Conference on Computer and Communications (ICCC), pp. 2800–2804, IEEE, 2016.

[24]. F. K. Lodhi, I. Abbasi, F. Khalid, and S. O. Hasan, F. Awwad, and S. R. Hasan, "A self-learning framework to detect the intruded integrated circuits," in 2016 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1702–1705, IEEE, 2016.

[25]. M. Xue, R. Bian, W. Liu, and J. Wang, "Defeating untrustworthy- thy testing parties: A novel hybrid clustering ensemble-based golden models-free hardware trojan detection method," IEEE Access, vol. 7, pp. 5124–5140, 2018.

[26]. D. Jap, W. He, and S. Bhasin, "Supervised and unsupervised machine learning for side-channel based trojan detection," in 2016 IEEE 27th International Conference on Application-specific Systems, Architectures and Processors (ASAP), pp. 17–24, IEEE, 2016.

[27]. Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris, "Silicon demonstration of hardware trojan design and detection in wireless cryptographic ics," IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 25, no. 4, pp. 1506–1519, 2016.

[28]. H. Zhang, J. Zhou, D. Gao, X. Wang, Z. Chen, and H. Wang, "Hardware Trojan detection based on ordered mixed feature gep," Security and Communication Networks, vol. 2021,2021

[29]. T. Hoque, J. Cruz, P. Chakraborty, and S. Bhunia, "Hardware ip trust validation: Learn (the untrustworthy), and verify," in 2018 IEEE International Test Conference (ITC), pp. 1–10, IEEE, 2018.

[30]. S. P. Moustakidis, K. G. Liakos, G. K. Georgakilas, N. Ske- topoulos, S. Seimoglou, P. Karlsson, and F. Plessas, "A novel holistic approach for hardware trojan detection powered by deep learning (hero),"

[31]. Z. Pan and P. Mishra, "Automated test generation for hardware Trojan detection using reinforcement learning," in Proceedings of the 26th Asia and South Pacific Design Automation Conference, pp. 408–413, 2021.

[32]. P. Dahiya and D. K. Srivastava, "Network intrusion detection in big dataset using spark," Procedia computer science, vol. 132, pp. 253–262, 2018.

[33]. N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on sdn based network intrusion detection system using machine learning approaches," Peer-to-Peer Networking and Applications, vol. 12, no. 2, pp. 493–501, 2019.

[34]. R. Singh, H. Kumar, R. K. Singla, and R. R. Ketti, "Internet attacks and intrusion detection system: A literature review," Online Information Review, 2017.

[35]. Sumaiya Thaseen, J. Saira Banu, K. Lavanya, M. Rukunud- din Ghalib, and K. Abhishek, "An integrated intrusion detection system using correlation-based attribute selection and artificial neural network," Transactions on Emerging Telecommunications Technologies, vol. 32, no. 2, p. e4014, 2021.

[36]. M. Li, "Application of cart decision tree combined with PCA algorithm in intrusion detection," in 2017 8th IEEE International Conference on Software Engineering and Service Science (ACCESS), pp. 38–41, IEEE, 2017.

[37]. Y. Zhou, G. Cheng, S. Jiang, and M. Dai, "An efficient intrusion detection system based on feature selection and ensemble classifier," arXiv preprint arXiv:1904.01352, 2019.S. Bhattacharya, P. K. R. Maddikunta, R. Kaluri, S. Singh, T. R.

[38]. Gadekallu, M. Alazab, U. Tariq, et al., "A novel PCA-firefly based boost classification model for intrusion detection in networks using GPU," Electronics, vol. 9, no. 2, p. 219, 2020.H.-T. Wang, J. Smallwood, J. Mourao-Miranda, C. H. Xia, T. D.

[39]. Satterthwaite, D. S. Bassett, and D. Bzdok, "Finding the needle in a high-dimensional haystack: canonical correlation analysis for neuroscientists," NeuroImage, vol. 216, p. 116745, 2020.

[40]. Z. Chen, S. X. Ding, T. Peng, C. Yang, and W. Gui, "Fault detection for non-gaussian processes using generalized canonical correlation analysis and randomized algorithms," IEEE Transactions on Industrial Electronics, vol. 65, no. 2, pp. 1559–1567, 2017.