National College of Ireland

Efficient Intrusion Detection for Smart Homes: Suricata and Machine Learning for Speed and Efficiency

MSc Research Project
MSc Cybersecurity

Navya Tumparthy
Student ID: 23101521

School of Computing
National College of Ireland

Supervisor: Khadija Hafeez

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……..…Navya Tumparthy……...

Student ID: ……23101521……

Programme: ……MSc Cybersecurity…… **Year:** …2023-2024.

Module: …… MSc Research Practicum part 2……

Supervisor: ……Khadija Hafeez……

Submission Due Date: ……12th August 2024……

Project Title: … Efficient Intrusion Detection for Smart Homes: Suricata and Machine Learning for Speed and Efficiency…….

Word Count: ……8608…… **Page Count**……19……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: …Navya Tumparthy…….

Date: …12th August 2024……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Efficient Intrusion Detection for Smart Homes: Suricata and Machine Learning for Speed and Efficiency

Navya Tumparthy
23101521

Abstract

Smart home devices and their integration with IoT has increased cyber-attacks significantly. Therefore, there is need for efficient Network Intrusion Detection systems (NIDS). Currently available IDS are not great because they produce number of false alarms and resource utilisation is high making them not suitable for smart homes where the computational power is limited. Hence, there is a need for Intrusion Detection Systems (IDS) that are quick in identifying attacks and use less computational resources. In this study, a hybrid machine learning model is integrated with Suricata to address the drawbacks of conventional IDS. Our model utilises the advantages of two algorithms, Random Forest (RF) for feature selection and LGBM (Lightweight Gradient Boost Model) for prediction. The models are trained on latest CICIoT2023 dataset and tested in a simulated smart home network by attack simulation. The enhanced model showed notable results especially with DDoS (Distributed Denial of Service), DNS tunnelling, and Mirai botnet attacks. Significant improvement in detection time and resource efficiency is observed. These studies provide notable advancement in IDS for real-time detections in resource constraint environments. Despite the success, the model needs performance improvement in few attack categories and analysis of commercial application is needed.

1 Introduction

Smart homes are becoming more popular these days and the reason for this is convenience. Humans like to have convenience and controllability on the devices. At present 360.72 million smart homes are observed as per statistics and is expected to increase by 86.74% by 2027. These smart homes have become smart because of the integration of IoT (Internet of Things). IoT is the concept where every device is connected to internet and smart home is just one of the applications of IoT (Alasmari and Alhogail, 2024). Smart homes have devices such as cameras, lights, fridge, thermostat, automatic door etc as in Figure 1 which are becoming IoT compatible. These devices communicate with each other by sharing information for effective working (Alghayadh and Debnath, 2020). Therefore, these devices are prone to risk as they deal with sensitive information over internet, and this motivated the researchers to analyse the security of smart home IoT devices. It is observed that the global security market of smart home has reached $4.3 billion from 2018 to 2022 which shows the importance of security (Alasmari and Alhogail, 2024). Providing security to smart homes is not simple as the security devices available in market are not compatible with IoT devices. Smart home does not have enough resources and even IoT devices are designed to work with less computational power.

Usually, security devices require significant resources to complete their latency-sensitive tasks. One more reason is that the devices in smart home are different from each other (In terms of hardware, software and protocols). Hence they demand specific security devices which makes it even more complicated to design computationally efficient security solution. Based on this we can say that traditional IDS cannot be used (Anthi et al., 2019).

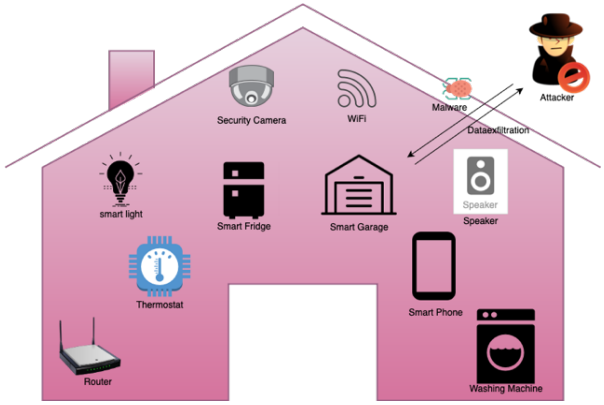

Figure 1: Smart Home

Traditional IDS are basically of two types, signature and anomaly based. Signature based IDS matches the network traffic patterns with already available signatures and alerts if there is a match thus having high accuracy. But there is no scope of detecting latest attacks as signatures will not be available. Whereas to detect new attacks anomaly-based IDS can be used, but they produce many false alarms. Additionally, traditional IDS are designed to work in high resource networks. Thus, an IDS should be designed specifically for smart home IoT network. This can be achieved with the help of ML (Machine Learning) techniques known to learn and predict with higher accuracies when trained with quality data. Employing these techniques can help with the diverse nature of IoT devices (Javed *et al.,* 2024). Different attacks like DDoS, botnet, unauthorized access etc., were analyzed to create an IDS for smart homes by past researchers. They aimed to train ML models on variety of datasets for higher accuracy and precision. But they did not address the computational drawbacks of smart homes. Also, they did not validate their work in real-time or simulated smart home networks.

In this paper the critical need for a better security solution in smart homes is addressed by integrating a hybrid ML model for a well-known open-source IDS Suricata. The algorithms are selected for better accuracy, speed and consume less resources in IoT environment. The main aim of this study is to use hybrid machine learning and implement it in simulated environment. In this study we answer the major concerns of smart homes by developing an affordable yet efficient model that can be used by security providers, smart home users and IoT manufacturers. Hybrid model using RF and LGBM, later integrating this with Suricata is proposed in this study after analysing the gaps in past research. The drawbacks of past research as in Rani et al., (2023), Khan and Sharma (2024) are addressed. Suricata is chosen for its higher accuracy and capability of handling network traffic (Andrew DeVito, 2024). Main objective of this research is to develop a lightweight model which is accurate, fast and computationally efficient. The successful implementation of this approach can provide a better security solution which will be adaptive and capable of identifying various cyber-attacks in the evolving smart homes. This research also helps the security researchers to emphasise more on the diversity of smart homes and to verify their models with real-time scenarios.

The basic problem with smart home devices is the heterogeneity meaning the hardware, software, protocols etc are different for each device. Thus, researchers identified the need for ML to solve the issue (Rani *et al.,* 2023). But, resource constraints of smart home are neglected. Algorithms were selected based on accuracy rather than its computational efficiency. On the other hand, there was no practical implementation of the developed IDS in real world nor in a simulated smart home network. Considering all these research gaps below mentioned research question and objectives are addressed in this study.

Research Question: "How can the integration of a hybrid machine learning model with Suricata enhances the accuracy, detection speed, and resource efficiency of network intrusion detection systems in a simulated smart home environment?"

Objectives:

- Training a hybrid ML model (RF and LGBM) using CICIoT2023 dataset.
- Trained model should be integrated with Suricata.
- Smart home should be simulated and monitored through ML integrated Suricata.
- Analyse time of detection and resource efficiency of Suricata-ML by attacking the smart home. Results should be compared for both Suricata and ML and with previous work to ensure ML took less time and resources for attack detection.

Therefore, in this study hybrid machine learning model, where layer 1 is trained using Random Forest algorithm known for its accuracy and features are extracted, layer 2 is trained using LGBM known to be computationally efficient, thus addressing the major concerns of IoT smart homes. Smart home network is simulated using Mininet. Suricata is configured to monitor and log the network traffic and ML model is integrated with it enabling real-time detection by analysing the network traffic.

Limitations:

1. Model trained on CICIoT2023 dataset meaning it might not work well for few attack types as the data in real world is vast.
2. Performance of model might vary in real-world as we tested it on simulated network.
3. Attacks performed during evaluation are considered as if the attacker knows the dataset.
4. Computational power and speed will be different in real-world.
5. Unable to perform few attacks as it is a simulated network.

Further sections of paper are organised as, first section explains the literature review conducted on variety of papers by analysing the IDS in smart home, noting the limitations and taking ideas. In the Research Methodology section selection of ML algorithms, datasets used, dataset processing and how exactly the model is trained and implemented in simulated network is presented. Under Design Specification section the architecture of the proposed model and its components will be explained. In Implementation, the integration of model with Suricata, simulated setup and its real-time detection capabilities will be provided. Evaluation section provides comprehensive details about the model's performance with respect to different metrics as well as speed and resource efficiency. Finally, Conclusion and Future Work gives summary of the model's performance in simulated network and scope for future improvements.

2 Related Work

2.1 DDoS detection using ML

In this study authors performed a comparative analysis on different standard and boosting algorithms to enhance IoT security from DDoS attacks. Models' accuracy was given paramount importance and based on the results boosting algorithms produced better accuracy which should be noted. Even though many datasets and algorithms were used nothing was done to enhance the resource efficiency (M, P and M, 2023). Similarly Das, Krishnamurthy and Das. (2022) also designed an IDS by comparative analysis. Instead of standard algorithms ensemble algorithms were used. Additionally, they used an IoT specific dataset and emphasised on false alarms. Feature selection techniques were put forth in this study. Ashraf and Elmedany (2021) did comprehensive research on DDoS attacks and types of algorithms that can be used. This paper gives detail knowledge about the past research, but no new ideas were implemented. In contrast to all the papers above, this paper used Deep Learning (DL) instead of just ML techniques for DDoS detection in IoT. This paper mentioned about the time efficiency and used BoT-IoT dataset, but resource constraints of IoT and real-time implementation is missing (Almaraz-Rivera, Perez-Diaz and Cantoral-Ceballos, 2022). In summary performance of different algorithms can be analysed from these researchers which helps us in choosing the right one for IoT environments.

2.2 SDN based solutions

Researchers of this study proposed an intrusion detection and prevention model using Software-Defined Networking (SDN) based deployment architecture specifically for smart homes. 'For better accuracy better feature detection is needed' was the motto of this paper and they evaluated the work using different models. Feature analysis was conducted from real smart home testbed and NSL-KDD dataset. Only accuracy was considered as evaluation metrics (Illy *et al.,* 2022). On the other hand, this study was focused on DDoS attack detection using SDN unlike Illy et al., 2022. They extracted the traffic during DDoS attacks and used standard algorithms for training and also employed SNORT an IDS for better detections (Garba *et al.,* 2024). De Melo *et al.* (2022), also used SDN to manage and analyse the home network flow called 'FamilyGaurd' emphasising dimensionality reduction using high end algorithms but are complex to deploy. Even though the work was comendable in all papers they used components which are computationaly intensive and can add extra time for detection, but using real data for training and common attacks on smart home were emphasised.

2.3 Deep Learning models

This paper emphasized on deep learning methods usage in smart homes and used only LSTM (Long short-term memory) as IDS. They proposed this model to ensure there is very less training time. There is neither model's performance evaluation in real world nor focused on computational efficiency. But they have used real time data for training, therefore the model is reliable (Azumah *et al.,* 2021). Unlike Azumah *et al*. (2021) research in this study focused on computational efficiency which was the major concern of smart homes. They used neural networks for feature extraction from a simulated network and used LSTM model for training.

They integrated firefly swarm optimization to reduce computational power and tested their model on public datasets (Alqahtani, 2022). But they evaluated for accuracy and F1-score rather than speed and computational power usage, thus not answering the main objective.

This paper addressed the impact of high dimensional network traffic on IDS. They utilized deep learning techniques for dimensionality reduction by ensuring the depth of the layers are optimized. Necessity of using algorithms with reduced time complexity in smart homes was emphasized alongside false positive reductions (Hu and Hu, 2023). However, they did not use IoT specific dataset and not tested their work in simulated smart home. But their model provided 60% accuracy for few novel attacks which should be noted. In contrary to the above presented work, the researchers in this paper used the combination of ML and DL methods. LSTM, KNN (k-nearest neighbors) and DT (Decision Tree) were used together as IDS in smart homes. More details about overfitting/underfitting of models is provided which are advantageous for my work. Additionally feature selection and hyper parameter tunings importance is emphasized (Butt et al., 2022). Despite the model is trained and tested on IoT dataset there was no focus on real time evaluation.

2.4 Machine Learning Approaches for IDS

This study provides the details about various types of IDS available, and which can be used better in smart home environments. Along with that the authors also explains about the algorithms available and applicable for smart home IDS (Prasad *et al.,* 2022). Apart from this knowledge nothing practical has been done hence this was used for initial understanding. In contrast (Khan and Sharma, 2024; Sarwar *et al.,* 2023; Khare and Totaro, 2020) tried to develop an IDS using standard ML as well as DL techniques using IoT specific datasets and achieved very high accuracies above 99%. However, the time and computational concerns are ignored. Additionally higher training and testing accuracies may indicate overfitting of the model, and they have not evaluated the performance in any real-time smart homes. Similar approaches were used by (Rahim *et al.,* 2023) in their study alongside a facial recognition model was established. But even their work did not answer the major concerns of smart home IoT devices.

In contrary to the all above mentioned past research (Rani *et al.,* 2023) and (Javed *et al.,* 2024) are the two researchers who focused mostly on the time efficiency of the model alongside the accuracy. Rani *et al.* (2023) captured traffic from real time smart home sensors for model training and used Raspberry Pi 4-based adversary for real time testing. Alongside accuracy and time efficiency, False Positive Rate (FPR) was also calculated. The model showed 99% accuracy meaning there might be overfitting, and the inference time was around 1.456 seconds. But the computational efficiency was not highlighted here, so to address this Javed *et al.* (2024) utilised many feature selection techniques like PCA and Chi-square alongside XGBoost providing tremendous results. Inference time of the better performing model was 0.003 seconds with FPR of 0.04%. But the computation power usage was way too high which was around 82.7%. Hence these two papers answered major concern's but still there is a room for improvement.

2.5 Signature and Behaviour-Based Models

Fairly a different approach was followed by Visoottiviseth *et al.* (2020) for IDS development in smart homes. The advantage of signature based, and behavior based are combined to develop a cost-effective model. But the CPU utilization of the model was 25.85% for few of the attacks and the model is not evaluated for the detection speed even though it was highlighted at the beginning of the paper. However, this motivated my work to utilize the benefits of signature-based model's accuracies.

2.6 Protocol-Specific Model

This paper was mainly focused on a single protocol used in IoT environment MQTT (Message Queuing Telemetry Transport). However, the important concepts such as over sampling and under sampling techniques for data balancing were covered. Additionally automatic feature engineering was used for improving models' performance and they have emphasized reduced time on classifying attacks (Alasmari and Alhogail, 2024). However, the times calculated were for training and testing rather than the time for predictions. Many things such as computational efficiency and implementation was also ignored. But other techniques were scrutinized for my work.

2.7 Specific Techniques and Applications

The authors followed unique approach to address smart home attacks. Li *et al.* (2022) proposed a solution of using User-Command-Chain (UCC), where the model is trained using the three attack types collected from the traffic of smart home testbed. Raspberry Pi within the HAN (Home Area Network) was implemented for evaluating the model and achieved 98.8% accuracy, but it was only for three attacks. Additionally, an email notification to customer was implemented. But the algorithms used were standard and no emphasis on resource constraints were made. In similar fashion novel techniques like using a self-learning model was developed by Wang, Yang and Weng. (2023) by highlighting the impotance of telemetry data in smart homes. But both used transofrmer based models making them more computationally intensive and time taking. The evaluation of model was generic and nothing new.

2.8 Feature selection

All the three authors have analyzed the importance of feature selection for models' accuracy and resource efficiency. Thus, this analysis provided us great insights on various feature selection techniques. Li, Hong and Yu. (2020) used Kolmogorov Smirnov (KS) test for selecting the features and implemented this technique in two layers of the proposed model. Similarly, Spanos *et al.* (2020) also extracted traffic and conducted time-series analysis for feature extraction using edge computing. PCA(Principle Component Analysis) feature selection was used. Even though every one used the general metrics for evaluation and ML/DL algorithms the feature importance method used by Gazdar (2022) was quite different and used public IoT dataset containing various attack types. In his work RF feature importance was used for better performance and proved ML outperforms DL, thus helping us to select RF and ML combination for my work.

Table 1: Summary of Literature

Authors (Year)	Aim of study	Models Used	Dataset	Findings	Drawbacks
M *et al.* (2023)	DDoS attacks in IoT	RF, ADA Boost, XG Boost and more	NSL-KDD, KDD-CUP 99, UNSW-NB15 CICDDOS-2019	Boosting algorithms performs better than standard algorithms	Dataset is not IoT sepcific
Das *et al.* (2022)	DDoS attacks in smart homes	Ensemble ML IDS	NSL-KDD, UNSW-NB15, CICIDS2017, DS2OS	Ensemble models outperforms single models with 99.5 F1-score, 0.05 false alarm rate	No IoT-specific dataset, high computational complexity and no implementation
Ashraf and Elmedany (2021)	DDoS detection in IoT	Review of various ML algorithms	-	Different ML algorithms efficiency on DDoS detections	No novel techniques, only reviews existing work
Almaraz-Rivera *et al.* (2022)	DoS & DDoS detection in IoT	Decision Tree and Multi-Layer Perceptron	Bot-IoT	99% accuracy, data balancing techniques, evaluating more than 1681 flows/sec	Not real-time implementation
Illy *et al.* (2022)	Enhanced IDPS with SDN	DT, KNN, RF, Bagging etc	NSL-KDD	Quality feature selection is highlighted	Increased computational power with SDN, high complexity
Garba *et al.* (2024)	DDoS detection in SDN-enabled smart homes	SVM, LR, DT, KNN	Real-time data	Real time detection facility and 99.57% accuracy with DT	High computational needs, limited attack focus
De Melo *et al.* (2022)	Anomaly detection in home networks	OCSVM, LOF, IF	Self-generated	Validated with Raspberry Pi, cost-effective	High computational power, not diverse datasets
Hu and Hu (2023)	Hybrid neural network IDS	Deep learning, fuzzy neural network (LSTM, CNN)	KDDCUP99	94% detection accuracy for DoS, illegal remote access	High initial computational power and non IoT data
Alqahtani (2022)	Hybrid IDS for smart homes	LSTM, FSO, CNN	CIDCC-15, UNSW-NB15, NSL-KDD	High accuracy, reduced computational overhead	Complex model, non-IoT-specific datasets
Azumah *et al.* (2021)	LSTM-based anomaly detection	LSTM	No dataset mentioned	Adaptability to new threats and high dimensional data	Resource-intensive, challenging to deploy
Butt *et al.* (2022)	Anomaly-based IDS for smart homes	KNN, DT, LSTM	CIC-IDS2022	High performance, feature selection importance	Lack of hyperparameter tuning details, high complexity
Rani *et al.* (2023)	Accurate and time-efficient IDS	LR, RF, XGB, LGBM	DS2OS	Great accuracy and time efficiency	Resource efficiency not calculated and not implemented the model
Javed *et al.* (2024)	Two-layered IDS for smart homes	XGBoost	Raspberry Pi-based	99.50% accuracy for DoS and Man in the Middle attack	Cloud dependency
Rahim *et al.* (2023)	Facial recognition for smart homes	LR, XGB, GBC, CNN	Labelled datasets	94% anomaly detection, 88% facial recognition	No real-time data, no focus on computational power
Prasad *et al.* (2022)	Survey on IDPS in smart environments	Various IDS and IPS techniques	-	Overview of IDS needs and attacks on IoT	Lacks implementation details
Sarwar *et al.* (2023)	IDS using multiple ML models	RF, DT, ADA, LSTM, ANN	UNSW BoT IoT	100% accuracy for RF, DT, ADA	Focus on just botnets
Khare and Totaro (2020)	Ensemble learning for anomaly detection	AdaBoost	DS2OS	Improved detection of multiple attack types	No real-world testing, lacks computational efficiency
Visoottiviseth *et al.* (2020)	Low-cost IDPS for smart homes	Signature and behaviour-based detection	-	Efficient detection, web monitoring	No detection speed, computational power
Khan and Sharma (2024)	Preventing unauthorized access in IoT	XGBoost, SNN, GaussianNB, LR	Aposemat IoT-23	99% accuracy for XGBoost, SNN	No CPU usage and implementation
Alasmari and Alhogail (2024)	MQTT-based IDS for IoT smart homes	Naïve Bayes, GLM, LR, RF, GBT	MQTT-IoT-IDS2020	100% accuracy with GLM	No CPU and speed aspects
Li *et al.* (2022)	Anomaly-based NIDS for smart homes	ML with UCC	Simulated test bed	98.8% accuracy	Not applicable for different devices

Wang *et al.* (2023)	Transformer-based NIDS for smart homes	Self-attention, Transformer model	Ton IoT	97.95% binary, 95.78% multi-class	High computational power, No real world evaluation
Li *et al.* (2020)	IDS for smart homes	Supervised ML, low-complexity feature selection	-	98.7% and 98.99% accuracy with 29 and 9 features	Limited attack types used for training
Gazdar (2022)	IDS for IoT-enabled smart homes	RF, DL models, feature selection	IoT/IIoT dataset	Device-specific model, effective feature selection	high initial computational power as DL used
Spanos *et al.* (2020)	Lightweight anomaly detection	Statistical and ML techniques, edge computing	-	Improved detection time with edge computing	Focus on time-series data, not real-world suitable

2.9 Research Niche

Significant gaps were identified from past research. Most of the study was focused on model performance. In few of the papers the IDS was trained for smart homes, but the datasets used were irrelevant. Algorithm selection was focused on accuracy and other ML metrics rather than the computational necessities. Most of the papers did not consider evaluating the model based on prediction time as it is important for the model to be quick enough in real-time. There is a lack of testing in simulated real-world IoT scenarios. However, few papers suggested the need of feature selection for lightweight models which can provide insights for my work. Thus, my research aims to use a hybrid ML model to address computational overhead and detection speed issues in IoT smart home environments. I will use a lightweight model utilizing the latest IoT-specific dataset CICIoT2023. Suitable feature selection methods for dimensionality reduction, will be employed. My work will replicate an IoT smart home network, integrating the ML model with Suricata for real-time intrusion detection aiding the signature-based detection and testing against attacks by evaluating computational requirements for real-time application.

3 Research Methodology

In this section, the structure of the research methodology is described. There are five main stages to the methodology; First stage is pre-processing the dataset, where data-cleansing and balancing is performed for model reliability and accuracy. In the second stage, the dataset is transformed with the help of feature selection for better performance using RF (Random Forest) algorithm. The dataset resulted from feature selection is passed to the third stage, namely the training and testing stage. In this stage, LGBM was used to train and test for better accuracy and speed. Trained model is integrated with Suricata an IDS to monitor traffic in simulated smart home in the fourth stage. In the final stage the implemented model will be evaluated for FPR (False Positive Rate), detection speed and computational power usage in smart home network through attack simulation.

3.1 CICIoT2023 Dataset

The dataset used in this paper is CICIoT2023 which is one of the latest datasets by Neto et al. (2023). The dataset is created by analysing the traffic of various IoT devices including the devices from smart home such as indoor cameras, smart lights, coffee maker, TV, speakers etc. While data collection many attacks were performed. So, there are total of 33 attack scenarios

considered in this dataset which can be broadly classified into 8 classes benign, Mirai, spoofing, brute force, web-based, recon, DoS, and DDoS attacks. The column of the dataset contains the features extracted from network traffic like flow_duration, protocol_type, header_legnth, duration, tcp flags, rate, variance, magnitude etc which are captured from network pcap files (Thereza and Ramli, 2023).

3.2 Dataset Preprocessing

The dataset was pre-processed in two stages. In first stage by using python scripts, we analysed the dataset for any empty/NA/Missing values and removed them. Later this dataset is again processed in such a way that the 33 attack types are transformed into 8 higher level classes. Then the count of 8 classes were printed to understand the balance of data. Counts of each class are DDoS: 9361472, DoS: 2227900, Mirai: 725551, Benign: 302896, Spoofing: 134176, Recon: 97110, Web-based: 6850 and BruteForce: 3590. As these classes show imbalance by using techniques like under sampling and oversampling we balanced it. To reduce the classes with higher counts RandomUnderSampler is used and to increase the classes with lower counts Synthetic Minority Over-sampling Technique (SMOTE) is used. Now the balanced dataset can be used for efficient ML training.

3.3 Machine learning Algorithms

Machine learning classification algorithms are of two types. Single classifiers use one algorithm whereas ensemble methods use many models and provide the results by combining the outputs. Hence we can say that ensemble classifiers are accurate than single classifiers. Multiple decision trees are combined for better accuracy in ensemble methods. There are two main types of ensemble methods: bagging and boosting. Bagging algorithms constructs all models simultaneously and reduce variance, while boosting algorithms reduce bias by building each model sequentially based on error of the previous model (Rani *et al.,* 2023). Therefore, ensemble models are used to reduce errors, hence in this paper two ensemble methods are used.
Random Forest: RF is a bagging method where it uses multiple decision trees. These trees are created by splitting the dataset into many subsets and the model is trained on all these. Advantages of RF are that it works well with high dimensional data, it is accurate as it uses many trees and avoids overfitting. Additionally, it is very good for feature selection as it uses different features in each subset of data and provides accurate feature importance. But it takes significant resources to train which is a drawback. (Rani *et al.,* 2023).
LGBM: LGBM is a histogram-based algorithm. It also uses decision trees but carefully considers the splits. Main advantage of LGBM is that it is lightweight algorithm and known to work well with large data. Accuracy is also major benefit of it. LGBM can be used for classification and decision-making problems as it is accurate. Even compared with other boosting algorithms it is good. (Rani *et al.,* 2023).

3.4 Feature Selection

Reducing features is very important as it helps with accuracy which was mentioned in past works. Also, by reducing features the burden on the algorithm can be reduced so the computational power will be reduced addressing our need. In this paper we used RF for feature

selection. As explained RF is known for its feature importance properties as it has built-in methodology for it. Therefore, by using this method we reduced the features from 46 to 20 before training the LGBM model. The idea behind choosing RF for feature selection is obtained from the work proposed by Gazdar (2022). Hence by following this approach we reduced the computational usage on the final model.

3.5 Training and Testing

Now the dataset is split into 70-30 ratio after balancing and pre-processing. During split stratification is used to ensure proper proportions of classes. By doing so we can ensure that training and testing are conducted on different data and the performance can be evaluated accurately and we can rely on model. Process followed is shown in Figure 2.

Phase 1: First RF model is trained and tested on dataset. While training this we measured accuracy by tuning few parameters like n_estimators, max_depth, min_samples_split, min_samples_leaf, boot_strap, random_state. Once the desired results are obtained the top 20 features are printed out and passed on to phase 2.

Phase 2: Similarly in phase 2 the top 20 features are used for training the LGBM model on 80% of data. Even in this stage for better accuracy and detection speed, parameter tuning is considered and is done on number of leaves, maximum depth, learning rate, number of estimators, and regularization parameters (L1 and L2) (Rani et al., 2023).

Cross-Validation: Cross validation is a technique where the dataset is divided into many splits or folds and these are used for training and testing the model. By doing so we can improve and understand the performance of the model. In addition to that we can know if the model is overfitting (Rani et al., 2023). In this paper we used StratifiedKFold method with 5 splits.

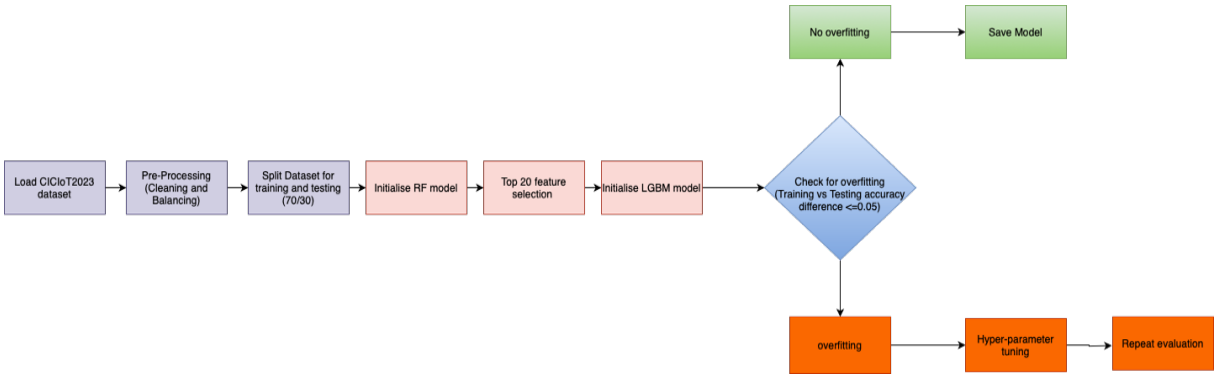

Figure 2: Model Training process

3.6 Evaluation Metrics

Different metrics are chosen to evaluate the model's performance. CICIoT2023 dataset is used to understand the efficiency of trained model. Detailed analysis is provided in evaluation section.

Classification Report: We included metrics such as accuracy, precision, recall, F1-score to analyse model's performance for each class.

- Accuracy: Accuracy is the calculation of correctly classified attacks to the total number of cases (Alasmari and Alhogail, 2024).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision: Percentage of accurate prediction of an attack over total number of samples predicted as attacks (Alasmari and Alhogail, 2024).

$$Precision = \frac{TP}{TP + FP}$$

- Recall: Percentage of accurate prediction of an attack over total number of actual attacks (Alasmari and Alhogail, 2024).

$$Recall = \frac{TP}{TP + FN}$$

- F1 Score: Combination of precision and recall which explains if the model has correctly classified malicious input while minimizing false positives and false negatives rates (Alasmari and Alhogail, 2024).

$$F1 - Score = 2(\frac{Precision * Recall}{Precision + Recall})$$

Confusion Matrix: It provides the detailed analysis on the correct and wrong predictions of every class. (Rani *et al.,* 2023).

ROC Curve and AUC: For every class Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) is found. These metrics are used for analysing the performance of classification models (Rani *et al.,* 2023).

Training and Testing Accuracy: To check if the model is overfitting we have analysed the accuracy for both testing and training data, and ensured the difference does not exceed 0.05.

Time taken for single prediction: During evaluation to check the detection speed we used python library 'time'. This is calculated for single prediction so that we will know if this model is fast enough.

Model Saving and Loading: The trained model is saved for smart home implementation.

3.7 Implementation and Integration with Suricata

Firstly, by using Mininet a smart home network is simulated and ensured the traffic of smart home contains all the necessary features. In next step Suricata a well-known IDS is installed and configured to monitor all network interfaces of smart home. Later, the Suricata is integrated with the saved model. This is achieved by extracting appropriate logs from Suricata and feeding those as features to the ML model for analysis and prediction. The detailed integration mechanism is explained in the Implementation section.

3.8 Evaluation and Analysis

The proposed model is evaluated in the simulated smart home network. This is achieved by simulating attacks on the smart home and the ability of the ML model along with Suricata to detect the attacks is analysed. The model is evaluated in two steps:

1. First the computational power used by Suricata alone to detect an attack is calculated.
2. Then the time and computational power used by ML to detect an attack is calculated.

Thus, by comparing the computational power used by Suricata and ML, we want to show that ML takes very less resources and is fast. And we evaluated the performance by comparing with previous work and proving the proposed model's applicability in smart homes.

4 Design Specification

In this section all components of the proposed model's architecture are put forth as in Figure 3. The ultimate goal is to develop an efficient model and integrate it with Suricata for network logs processing and detect attacks on smart home. This section provides detail explanation about the hardware and software of each component. Along with that the experimentation setup such as Virtual Machines (VM's) configurations, smart home simulation will be explained.

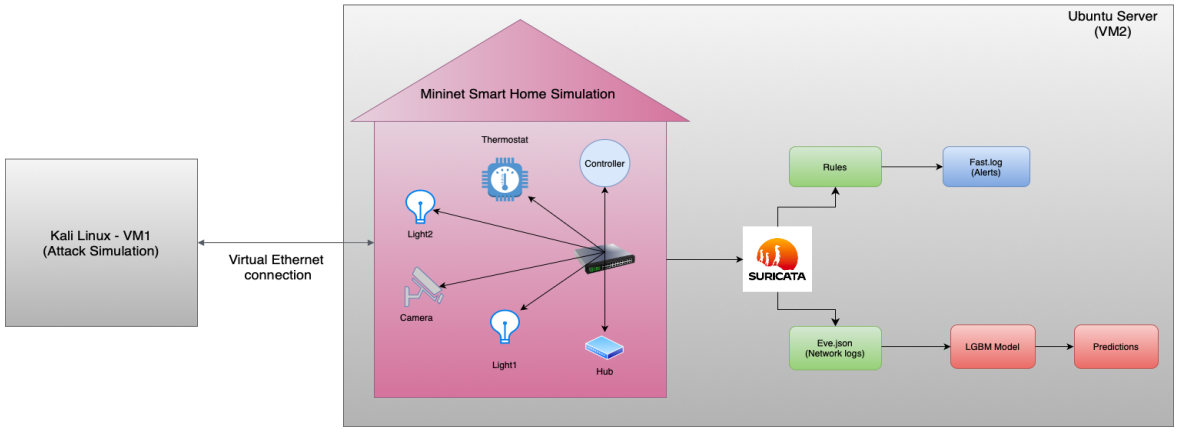

Figure 3: Architecture Diagram

Host Machine: Mac M1 Pro

MacBook Pro M1 was used for complete solution. It is of 8-core processor, 8GB RAM, and 256GB SSD, running macOS Sonoma version 14.5. Anaconda Navigator (v2.0.4) was installed for python programming, and it comes with Jupyter notebook which is used for data pre-processing and cleaning. Later, VMware Fusion (v13.5.0) is installed for VM's setup.

4.1 Dataset Preprocessing

Jupyter Notebook (v6.3.0) is used for dataset cleaning and balancing. Many libraries were used for this such as: OS[1] a built-in library of python is used for accessing environment variables of the host machine; Glob[2] a built-in library which is used for matching paths to access dataset files; Pandas[3] (v1.2.4) is used for dataset manipulation such as cleaning, processing etc.; Joblib[4] (v1.4.0) is used for saving the model; Imblearn[5] (v0.12.3) is used for data balancing (imblearn.under_sampling, imblearn.over_sampling.SMOTE).

4.2 Model Training

[1] https://docs.python.org/3/index.html
[2] https://docs.python.org/3/index.html
[3] https://pypi.org/project/pandas/
[4] https://joblib.readthedocs.io/en/stable/
[5] https://imbalanced-learn.org/stable/

Google Colab is a service which provides free access to computing resources like GPUs and TPUs. It is similar to Jupyter Notebook and can be used for ML (research.google.com, 2023). As this does not need any prior setup, we used this for model training. Libraries used during the model training were: Pandas (v2.0.3) for dataset manipulation; scikit-learn[6] (v1.2.2) for model building (RF), training, cross-validation; lightgbm[7] (v4.1.0) for importing the LGBM algorithm and training; Joblib (v1.4.2) for model saving and pipelining; seaborn[8] (v0.13.1) for creation of heatmap; matplotlib[9] (v3.7.1) for graphical representation of models performance; time[10] (built-in) is used to calculate time taken for prediction of single sample.

4.3 Virtual Machine Setup

VMware Installation: Installed VMware (v13.5.0) on host machine to have two VM's for smart home and attacker machine.

4.3.1 VM1: Kali Linux

To attack the smart home network Kali Linux was installed, as it comes with many tools which are used for attacking. We installed few more tools for attacking. Nmap[11] (v7.94SVN) is used for initial scan. Hydra[12] (v9.5) is installed for performing Brute-Force attacks. Hping3[13] (v3.0.0-alpha-2) is used to perform DDoS attacks on smart home via TCP flood as well as ICMP flood. Iodine[14] (v0.7.0) is installed for DNS tunnelling attacks.

Network Configuration: A Static IP is configured for Kali Linux for consist experimentation. Additionally, a route has been added to ensure Kali Linux can attack simulated smart home as the simulation was done using Mininet inside a different VM.

4.3.2 VM2: Ubuntu Server

The Ubuntu Server is chosen for the installation of Mininet a known network simulator and Suricata is also installed for monitoring smart home network. Ubuntu server is used as it is a light weight and easy to configure operating system, hence it was chosen for experimentation and lab setup.

Tools Installed:

- **Suricata (version 7.0.6):** It is an open-source IDS, IPS and network security monitoring engine which performs good. When compared with its competitors it is a lightweight and fastest system (Andrew DeVito, 2024).
- **Mininet (version 2.3.0):** Mininet is a network emulator that helps building virtual network including hosts, switches, controllers, and links. Standard Linux network software is used by Mininet hosts. OpenFlow is supported by its switches for Software-Defined Networking and flexible routing. It helps doing everything in a single PC or laptop (mininet.org, 2022).

[6] https://scikit-learn.org/stable/getting_started.html
[7] https://lightgbm.readthedocs.io/en/stable/
[8] https://seaborn.pydata.org
[9] https://matplotlib.org
[10] https://docs.python.org/3/index.html
[11] https://nmap.org/book/toc.html
[12] https://hydra.cc/docs/intro/
[13] https://linux.die.net/man/8/hping3
[14] https://gist.github.com/nukeador/7483958

Network Simulation: The smart home network is simulated in Mininet using python programming where 5 smart home devices, two lights, one camera, one thermostat and one hub were simulated as in Figure 3. A virtual ethernet connection was established between Mininet and Ubuntu host to ensure the connectivity. Additionally, static IP is assigned to the Ubuntu Server for consistency and virtual ethernet routes are configured for smart home connectivity.

5 Implementation

This section explains the detailed steps of model implementation along with the integration of ML with Suricata.

5.1 Suricata

Suricata is an open-source NIDS. It can act as IPS (Intrusion Prevention System) and network security monitoring engine based on its application. It uses multi-threaded architecture meaning it can process multiple tasks together making it faster in performance. It works by analysing the network packets and also it can be used for extracting logs, pcap files etc which will be useful for ML integration. (Andrew DeVito, 2024).

5.2 YAML Configuration

Suricata should be configured to monitor the smart home network and rules should be updated for signature-based detection. To achieve this suricata.yaml file should be configured. suricata.yaml file is the main file of Suricata, containing all the settings that needs to be configured as per the need (docs.suricata.io, 2024). Therefore, suricata.yaml file is modified to monitor each interface of the smart home under pcap section of the file. This ensures that network packets for these interfaces are monitored and captured for malicious activity verification. Packets captured are matched with the signature base for alerting.

5.3 Integrating the ML Model with Suricata

The integration of the trained model is conducted in two steps. In the first step we have configured Suricata to extract the necessary logs later these logs are passed on to model for prediction.

suricata.yaml configuration: This file contains multiple sections which can be modified based on the users need. So we have modified few sections of this file to extract the logs based on the 20 features used for model training. These changes should be done manually and if we perform these changes all the network logs will be extracted into a file named eve.json. These logging changes should be performed under 'outputs' section of file (docs.suricata.io, 2024). All these 20 features are added by analysing the CICIoT2023 dataset. Few traffic flow logs that we added in 'outputs' section are flow_duration, http logs, dns logs, tls logs, smtp, mqtt and more which will be logged under eve.json.

Model Deployment: To deploy the model we have saved the model during training. The logs extracted from eve.json are parsed and transformed into dataframes for each event and will be passed to model for predictions.

eve.json Parsing: Python programming was used to continuously monitor eve.json log file generated by Suricata in order to integrate ML model. Eve.json file was parsed, which has comprehensive network information captured as stated in suricata.yaml configuration, to extract the appropriate features for the model. To ensure real-time monitoring, only newly generated logs were handled. Watchdog library was used to monitor the real time logging.
Feature extraction: First a python function was written to get the necessary features. This function works to analyse each line of eve.json file and extract features such as flow time, protocol type, header length, and different flag counts. Other features like magnitude, radius, covariance etc are calculated dynamically. Therefore, by doing this all features necessary for models prediction were extracted.
Passing features to the Model: After the extraction of features from eve.json they were arranged into pandas dataframes and used as input for the pre-trained ML model which was saved using joblib library during the training. The collected features were given as inputs to the model for prediction for every new entry in the log file.
Prediction Logic: As the model is trained it will predict the attacks based on the features extracted from the traffic captured in eve.json file. Additionally python libraries were used to predict time taken for predicting each log entry. This was calculated by analysing the time intervals before and after the model's prediction function was called. Both the prediction and time taken for prediction were printed together as the output. This real-time monitoring helps to identify suspicious activities without fail by ML model.

5.4 Tools and Languages Used

Several tools and libraries were used in the process of integrating the saved model with the Suricata. We have used python programming for the integration and to do the job of eve.json parsing. Detailed list of tools and the purpose of use are; Json[15] (built-in) library is used for parsing the eve.json file generated by Suricata and extract the features in the format expected my ML; time (built-in) library provides various time related functions and we have used this to calculate the time taken for prediction for real-time analysis; Joblib (v1.4.2) is used to load the saved model and use it for real-time prediction; NumPy[16] (v1.26.4) is used for computing the statistical features from the extracted features; Pandas (v2.2.2) is used for creation of data frames from the extracted features before feeding them to ML model; datetime[17] (built-in) is used to calculate the durations and times from the log entries, such as calculating network flows; Watchdog[18] (v4.0.1) is used to detect new entries in log files and trigger processing, parsing and predictions. Lightgbm (v4.4.0) is used for model working.

6 Evaluation

The model is tested in two stages. Firstly, it is evaluated using CICIoT2023 dataset for the general ML performance metrics such as accuracy, precision, F1-Score etc. In the next stage the model is tested for its real-time applicability after integrating it with Suricata by attacking

[15] https://docs.python.org/3/library/json.html
[16] https://numpy.org/doc/stable/user/absolute_beginners.html
[17] https://docs.python.org/3/library/datetime.html
[18] https://pypi.org/project/watchdog/

smart home network. In this stage we evaluated detection time and computational efficiency. During the training the dataset is split into 70-30 and the trained model is evaluated on 30% unseen data.

Confusion matrix: The results of confusion matrix are shown in Figure 4. Based on the results we can say that the model is performing quite good as the dark blue instances in the matrix shows the correct prediction numbers and remaining values explains the wrong predictions. Based on the data it can be said that DDoS, DoS and Mirai classes have very less mis-predictions. But spoofing, Recon and Benign classes need little improvement.

ROC AUC Curve: The area under the curve explains the accuracy for each class and it can be seen that the accuracy is 100% for DDoS, DoS and Mirai classes as in Figure 4. Even the other classes show higher accuracies which is greater than 90%. Thus, based on this data we can say that the overall performance of the model is quite good.

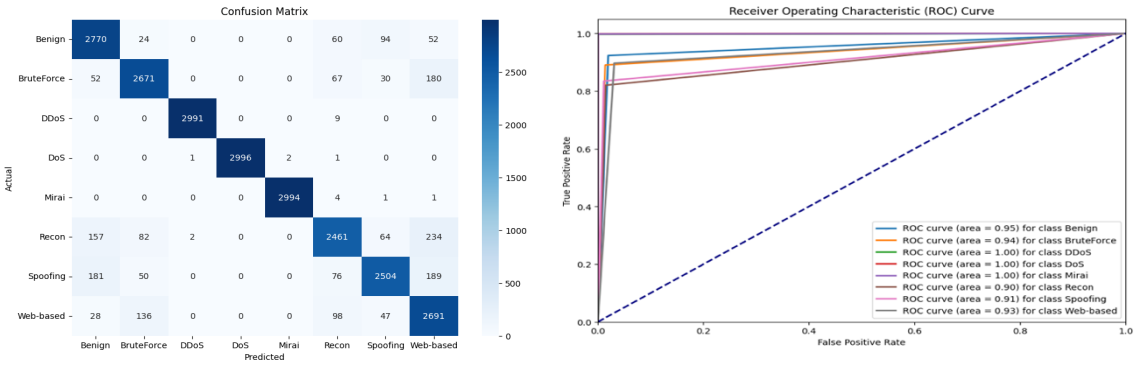

Figure 4: Confusion Matrix and ROC-AUC Curves

Classification Report: The training and testing values for precision, recall and F1-score are extracted and represented in a table while evaluating the model on test data as in Table 2. All three parameters are showing 100% for DDoS, DoS and Mirai attacks. Even the other classes are showing better values which is above 80%. In addition to this the overall accuracy of the model was evaluated which was 91.99% for testing data and 95.44% for training data which means the models accuracy is also good and it is not overfitting as the difference between accuracies is not greater than 5%. But we can observe that there is still need of improvement in few classes..

Table 2: ML metrics for each attack class

Class	Precision		Recall		F1-score	
	Training	Testing	Training	Testing	Training	Testing
Benign	0.91	0.87	0.95	0.92	0.93	0.90
Brute Force	0.95	0.90	0.95	0.89	0.95	0.90
DDoS	1.00	1.00	1.00	1.00	1.00	1.00
DoS	1.00	1.00	1.00	1.00	1.00	1.00
Mirai	1.00	1.00	1.00	1.00	1.00	1.00
Recon	0.95	0.89	0.89	0.82	0.92	0.85
Spoofing	0.96	0.91	0.89	0.83	0.93	0.87
Web-based	0.88	0.80	0.96	0.90	0.92	0.85

Error Rate: The difference between predicted and actual values is obtained from Mean absolute error (MAE). Square difference between predicted and actual values is mean squared error (MSE) and the average value of MSE is root mean squared errors (RMSE). These calculations help analyse the error-rates of the model and it should be less for better accuracy (Rani *et al.,* 2023). These values are very less for the proposed model as shown in the Table 3 explaining the models efficiency.

Table 3: Error-Rate

Algorithm	MAE	MSE	RMSE
LGBM	0.377	0.377	0.1943

TPR vs FPR: True positive rate (TPR) and false positive rate (FPR) explains the rate of detections. Another name for TPR is sensitivity and recall (Rani *et al.,* 2023). Table 4, shows the results of TPR and FPR. FPR is 0.0363 which is less.

Table 4: TPR vs FPR

TPR	0.9523
FPR	0.0363

Prediction Time: The time taken by model for single prediction was calculated during the model evaluation. By doing this we understood the speed of detection as this helps in real-world implementation. Faster detecting IDS are always better for security so we selected this metric. The average time taken for making a single prediction per sample is 0.004501 seconds, thus making it suitable for real-time detections and efficient for implementation.

To ensure the models real world applicability we implemented it in a smart home network for network monitoring. By performing different attacks, the model's capability of detection is analysed. Additionally, the speed of detection and resource efficiency were calculated for each attack. Computational usage is calculated using both code as well as monitoring the resources in ubuntu server.

6.1 Experiment 1: DDoS attack

First experiment was conducted by performing DDoS attack on smart home from Kali VM. DDoS is chosen as it is well-known attack on smart homes in the recent days and many researchers focused on this particular attack (Ashraf and Elmedany, 2021). During the attack we observed that the ML took 0.001219 seconds for detection. The CPU usage of ML is 1% and Suricata used 3%. Even the memory usage is lesser in ML which is 159.44 MB and Suricata used 873.25MB.

6.2 Experiment 2: DNS tunnelling attack

Second experiment was DNS tunnelling on one of the smart home devices. The time taken for prediction is 0.0056 seconds. The CPU utilisation of ML and Suricata is 4.1% and 21.4% respectively. Memory usage is 873 MB by Suricata and 159 MB by ML.

6.3 Experiment 3: Mirai Botnet

Third attack opted was Mirai botnet as this is also one of the popular attacks on IoT devices. Instead of direct attacking we downloaded a pcap file containing the traffic of Mirai attack from the Aposemat Project 2023 (Stratosphere IPS, 2023). From this pcap file we extracted the features that should be needed by ML for prediction. We analysed the CPU usage and time taken for detection. For this detection ML took 0.0064 seconds. Memory and CPU usage is 0.03 MB and 28.3% respectively. From this it can be analysed that the model took little high amount of CPU for this experiment. This is all done in google colab.

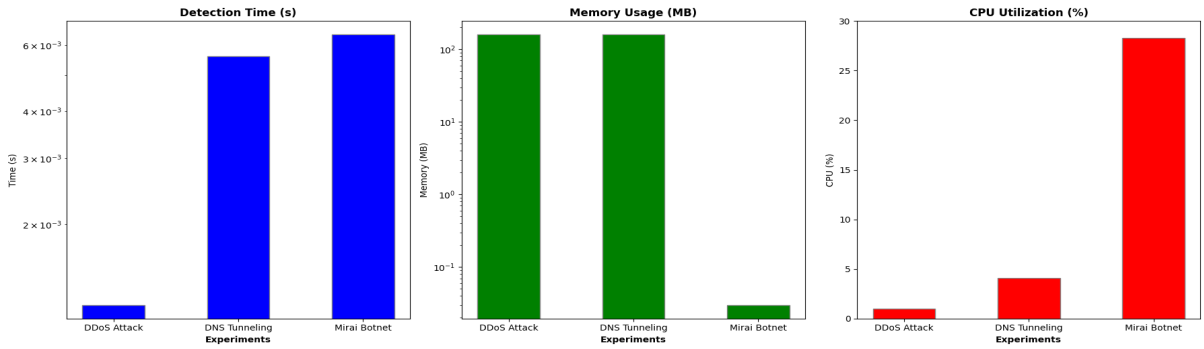

Figure 5: speed, memory and CPU utilization for experiments

6.4 Discussion

In this section the detailed discussion of the results will be provided. We aim to give complete idea on the results obtained by comparing the results with the previous papers and also with our own work. Both the strengths and weaknesses of the proposed model will be explained.

Strengths: The model showed 91.99% accuracy for unseen data which is good when it is seen on its own. As we compared the training and testing accuracies we can say that the model is not overfitting and is performing good. This explains the model's generalisability. The proposed model is tested in simulated smart home network by real-time attacks, and it has shown promising results as explained in the above sections. In addition, we have also tested the model by using a pcap file which is selected from entirely different project and environment. DDoS attack shows that the proposed model used 4% of CPU together Suricata-ML which is very less and making this suitable for smart homes. The time taken for predictions is also very less and the maximum time taken is in Mirai attack which is 0.0064 seconds. When compared with the past research conducted by Rani et al., (2023) where LGBM model is used the time taken for predictions was 1.465 seconds meaning our model is better. We also compared the MAE and RMSE from Rani et al., (2023) even these results are better in our model. Even the Suricata's resource usage is very less which is not more than 25%. The other major benefit is the advantage of signature-based detection for higher accuracies. FPR of the proposed model is 0.0363 and it is around 0.04 by the best model proposed by Javed et al., (2024). The models proposed by Javed et al., (2024) took more than 80% of CPU but our models CPU utilisation along with Suricata has not exceeded 30%. Based on all these values it can be said that the proposed model has better accuracy, speed of detection and resource efficiency making it well suitable for real world application.

Weaknesses: Accuracy of most of the models from the past research was nearly 99%. So, when compared the model presented in this paper has showed less accuracy of 91.99%. In addition to that few classes were not performing well considering the precision, F1-Score and Recall like Brute Force, Benign etc. Thus, there is a need to improve all these aspects. Even though FPR is better than one of the previous papers it is not sufficient. It is always good to have lesser FPR rates for practical implementations. The computational power usage for Mirai botnet attack was little high at 28.3% as shown in Figure 5 and it should be reduced. The model is tested on a simulated smart home network, but it is not sufficient as there will be limitations in simulated networks. The model is trained on single dataset which is not sufficient as it will not contain all the attacks in the world. So, there are high chances that the model might fail in few attack scenarios.

7 Conclusion and Future Work

The primary question addressed in this study is on how we can use hybrid machine learning with Suricata to enhance the accuracy, detection speed and resource efficiency of an IDS. To answer this question, we have established few objectives like training an ML model, integrating it with Suricata and using this as IDS for a simulated smart home. Later we tested the model by simulating attacks on the smart home network. The developed model was almost successful in answering the established question. The ML model integrated with Suricata demonstrated notable accuracy, speed and resource efficiency. This was concluded through extensive evaluation including real-time attack simulations. Model evaluation has proved that it can predict most of the classes with accuracy above 95% and the detection speed was also very less during the attack simulation. In addition, resource utilisation was very less making the model suitable for smart homes. Compared to Suricata the model alone added very less computational power and memory.

Despite the success there are few limitations which can be observed for the proposed model. The model seems to be less accurate for few classes. Furthermore, the overall accuracy of the model should also need betterment. The model is tested in the simulated network with attack simulations and also for malware pcap files, but there is need to evaluate model for more diverse datasets and attacks to understand its capability. Additionally, the network simulation should be of more diverse and there is a room to add more devices and protocols for better testing. Therefore, in future I would like to implement the model in an advanced simulated smart home or real smart home network. In addition to that, I would test the model's performance on different datasets and on different types of attacks for generalizability. Adaptive learning techniques will be implemented on this model to make it more efficient and robust over time. This model can also be commercialized by creating a user interface and putting the entire solution in a package for smart home security system. This includes communicating with latest smart home companies and developing a simple easy to use security solution for non-technical users. In summary, the proposed model has provided significant improvement in IDS detection capabilities in smart homes making it more viable in resource restrictive environments with no cost.

References

Alasmari, R and Alhogail, A.A. (2024) ‘Protecting Smart-Home IoT Devices From MQTT Attacks: An Empirical Study of ML-Based IDS’, *IEEE Access,* 12, pp. 25993–26004. doi: 10.1109/ACCESS.2024.3367113

Almaraz-Rivera, J.G., Perez-Diaz, J.A. and Cantoral-Ceballos, J.A. (2022) ‘Transport and Application Layer DDoS Attacks Detection to IoT Devices by Using Machine Learning and Deep Learning Models’, *Sensors*, 22(9), pp. 3367. doi: 10.3390/s22093367

Alghayadh, F. and Debnath, D. (2020) ‘A Hybrid Intrusion Detection System for Smart Home Security’, in *2020 IEEE International Conference on Electro Information Technology (EIT)*, Chicago, IL, USA, July 2020, pp. 319–323. doi: 10.1109/EIT48999.2020.9208296

Alqahtani, A.S. (2022) ‘FSO-LSTM IDS: hybrid optimized and ensembled deep-learning network-based intrusion detection system for smart networks’, *The Journal of Supercomputing*, 78(7), pp. 9438–9455. doi: 10.1007/s11227-021-04285-3

Andrew DeVito, (2024) *Suricata vs Snort: A Comprehensive Review.* Available at: https://www.stationx.net/suricata-vs-snort/ [Accessed 15 April 2024].

Anthi, E., Williams, L., Slowinska, M., Theodorakopoulos, G. and Burnap, P. (2019) ‘A Supervised Intrusion Detection System for Smart Home IoT Devices’, *IEEE Internet Things Journal,* 6(5), pp. 9042–9053. doi: 10.1109/JIOT.2019.2926365

Ashraf, A. and Elmedany, W.M. (2021) ‘IoT DDoS attacks detection using machine learning techniques: A Review’, in *2021 International Conference on Data Analytics for Business and Industry (ICDABI)*, Sakheer, Bahrain, 25 October 2021, pp. 178–185. doi: 10.1109/ICDABI53623.2021.9655789

Azumah, S.W., Elsayed, N., Adewopo, V., Zaghloul, Z.S. and Li, C. (2021) ‘A Deep LSTM based Approach for Intrusion Detection IoT Devices Network in Smart Home’, in *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, New Orleans, LA, USA, 14 June 2021, pp. 836–841. doi: 10.1109/WF-IoT51360.2021.9596033

Butt, N., Shahid, A., Qureshi, K.N., Haider, S., Ibrahim, A.O., Binzagr, F. and Arshad, N. (2022) ‘Intelligent Deep Learning for Anomaly-Based Intrusion Detection in IoT Smart Home Networks’, *Mathematics*, 10(23), pp. 4598. doi: 10.3390/math10234598

Das, R.R., Krishnamurthy, B. and Das, S. (2022). ‘Securing IoT devices using Ensemble Machine Learning in Smart Home Management System’, in *2022 IEEE Symposium Series on Computational Intelligence (SSCI).* Singapore, Singapore, 4 December 2022, pp. 915–922. doi: /10.1109/SSCI51031.2022.10022068

De Melo, P.H.A.D., Miani, R.S. and Rosa, P.F. (2022) ‘FamilyGuard: A Security Architecture for Anomaly Detection in Home Networks’, *Sensors*, 22(8), pp. 2895. doi: 10.3390/s22082895

docs.suricata.io. (2024) *12.1. Suricata.yaml — Suricata 8.0.0-dev documentation*. Available at: https://docs.suricata.io/en/latest/configuration/suricata-yaml.html [Accessed 15 June 2024].

Garba, U.H., Toosi, A.N., Pasha, M.F. and Khan, S. (2024) 'SDN-based detection and mitigation of DDoS attacks on smart homes', *Computer Communications*, 221, pp. 29–41. doi: 10.1016/j.comcom.2024.04.001

Gazdar, T. (2022) 'A New IDS for Smart Home based on Machine Learning', in *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*. Al-Khobar, Saudi Arabia, 4 December 2022, pp. 393–400. doi: 10.1109/CICN56167.2022.10008310

Hu, R. and Hu, X. (2023) 'An Intrusion Detection Method for IoT-Based Smart Home Based on Hybrid Neural Network', in *2023 IEEE 13th International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*. Qinhuangdao, China, 11 July 2023, pp. 823–829. doi: 10.1109/CYBER59472.2023.10256462

Illy, P., Kaddoum, G., Kaur, K. and Garg, S. (2022) 'ML-Based IDPS Enhancement With Complementary Features for Home IoT Networks', *IEEE Transactions on Network and Service Management*, 19(2), pp.772–783. doi: 10.1109/TNSM.2022.3141942

Javed, A., Ehtsham, A., Jawad, M., Awais, M.N., Qureshi, A., and Larijani, H. (2024) 'Implementation of Lightweight Machine Learning-Based Intrusion Detection System on IoT Devices of Smart Homes', *Future Internet*, 16(6), pp. 200. doi: 10.3390/fi16060200

Khan, A. and Sharma, I. (2024) 'SSH-BA: Safeguarding Smart Homes with Intelligent IoT Behavior Analysis Using Classification Techniques', in *2024 21st Learning and Technology Conference (L&T)*. Jeddah, Saudi Arabia, 15 January 2024, pp. 296–301. doi: 10.1109/LT60077.2024.10469565

Khare, S. and Totaro, M. (2020) 'Ensemble Learning for Detecting Attacks and Anomalies in IoT Smart Home', in *2020 3rd International Conference on Data Intelligence and Security (ICDIS)*. South Padre Island, TX, USA, June 2020, pp. 56–63. doi: 10.1109/ICDIS50059.2020.00014

Li, T., Hong, Z. and Yu, L. (2020) 'Machine Learning-based Intrusion Detection for IoT Devices in Smart Home', in *2020 IEEE 16th International Conference on Control & Automation (ICCA)*. Singapore, 9 October 2020, pp. 277–282. doi: 10.1109/ICCA51439.2020.9264406

Li, X., Ghodosi, H., Chen, C., Sankupellay, M. and Lee, I. (2022) 'Improving Network-Based Anomaly Detection in Smart Home Environment', *Sensors*, 22(15), pp. 5626. doi: 10.3390/s22155626

M, A., P, S. and M, K. (2023) 'Comparative Evaluation on Various Machine Learning Strategies Based on Identification of DDoS Attacks in IoT Environment', in *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*. Coimbatore, India, 17 March 2023, pp. 1814–1821. doi: 10.1109/ICACCS57279.2023.10112877

mininet.org (2022) *Mininet: An Instant Virtual Network on your Laptop (or other PC)*. Available at: http://mininet.org [Accessed 15 July 2024]

Neto, E.C.P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R. and Ghorbani, A.A. (2023) 'CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment', *Sensors*, 23(13), pp.5941. doi: 10.3390/s23135941

Prasad, S., Alamuru, S., S, A. and Alamuru, S. (2022) 'Intrusion Detection and Prevention Systems in Smart Environments – A Survey', in *2022 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*. Bangalore, India, 23 December 2022, pp. 1–5. doi: 10.1109/smartgencon56628.2022.10084309

Rahim, A., Zhong, Y., Ahmad, T., Ahmad, S., Pławiak, P. and Hammad, M. (2023) 'Enhancing Smart Home Security: Anomaly Detection and Face Recognition in Smart Home IoT Devices Using Logit-Boosted CNN Models', *Sensors*, 23(15), pp. 6979. doi: 10.3390/s23156979

Rani, D., Gill, N.S., Gulia, P., Arena, F. and Pau, G. (2023) 'Design of an Intrusion Detection Model for IoT-Enabled Smart Home', *IEEE Access*, pp. 1–1. doi: 10.1109/ACCESS.2023.3276863

research.google.com. (2023) . Google Colab. Available at: https://research.google.com/colaboratory/faq.html#:~:text=Colab%20is%20a%20hosted%20Jupyter [Accessed 13 July 2024]

Sarwar, N., Bajwa, I.S., Hussain, M.Z., Ibrahim, M. and Saleem, K. (2023) 'IoT Network Anomaly Detection in Smart Homes Using Machine Learning', *IEEE Access*, 11, pp. 119462–119480. doi: 10.1109/ACCESS.2023.3325929

Spanos, G., Giannoutakis, K.M., Votis, K., Viano, B., Augusto-Gonzalez, J., Aivatoglou, G. and Tzovaras, D. (2020) 'A Lightweight Cyber-Security Defense Framework for Smart Homes', in *2020 International Conference on Innovations in Intelligent Systems and Applications (INISTA)*. Novi Sad, Serbia, August 2020, pp. 1–7. doi: 10.1109/INISTA49547.2020.9194689

Stratosphere IPS. (2023) *Aposemat Project: IoT Malware Datasets*. Available at: https://www.stratosphereips.org/datasets-iot [Accessed 22 Jul. 2024].

Thereza, N. and Ramli, K. (2023) 'Development of Intrusion Detection Models for IoT Networks Utilizing CICIoT2023 Dataset', in *3rd 2023 International Conference on Smart Cities, Automation and Intelligent Computing Systems, ICON-SONICS 2023*. Bali, Indonesia, 6 December 23, pp. 66-72. doi: 10.1109/ICON-SONICS59898.2023.10435006

Visoottiviseth, V., Sakarin, P., Thongwilai, J. and Choobanjong, T. (2020) 'Signature-based and Behavior-based Attack Detection with Machine Learning for Home IoT Devices', in *2020 IEEE REGION 10 CONFERENCE (TENCON)*. Osaka, Japan, 16 November 2020, pp. 829–834. doi: 10.1109/TENCON50793.2020.9293811

Wang, M., Yang, N. and Weng, N. (2023) 'Securing a Smart Home with a Transformer-Based IoT Intrusion Detection System', *Electronics*, 12(9), pp. 2100. doi: 10.3390/electronics12092100