

AI-Based Detection of DDoS Attacks in IoT Smart Home Environments

Joseph Oluwasanmi Owolabi

240249395

Submitted in accordance with the requirements for the degree of
Master of Science in Computer Science

Under the Supervision of

Dr. Soonleh Ling

York St John University- London Campus

Department of Computer Science

January 2026

Declaration

I can confirm that the work submitted is my own and that appropriate credit has been given where reference has been made to the work of others.

This copy has been supplied on the understanding that it is copyright material. Any reuse must comply with the Copyright, Designs and Patents Act 1988 and any licence under which this copy is released.

© 2026 York St John University

The right of the Candidate's Name to be identified as the Author of this work has been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

Self-declaration on Academic Integrity:

I have read and understood the Academic Misconduct statement .	Tick to confirm <input checked="" type="checkbox"/>
I have read and understood the Generative Artificial Intelligence use statement .	Tick to confirm <input checked="" type="checkbox"/>
I am satisfied that I have met the Learning Outcomes of this assignment (please check the Assignment Brief if you are unsure)	Met <input checked="" type="checkbox"/>

Self-Assessment – If there are particular aspects of your assignment on which you would like feedback, please indicate below.

Optional for students

Suggested prompt questions-

How have you developed or progressed your learning in this work?

What do you feel is the strongest part of this submission?

What feedback would you give yourself?

What part(s) of this assignment are you still unsure about?

Abstract

The spread of Internet of Things (IoT) devices in the operation of smart home environments has brought about notable convenience in homes but at the same time has introduced new cybersecurity challenges. One of the most critical threats in this regard is the Distributed Denial of Service (DDoS) attacks, which exploit the resource constraints and connectivity of IoT devices to disrupt normal network operations. This dissertation will look to examine the application of Artificial Intelligence (AI), especially machine learning algorithms, in proactively detecting DDoS attacks within IoT-based smart homes.

This study begins with an in-depth review of the existing intrusion detection systems and points out the limitations of traditional rule-based approaches of detecting intrusions in a dynamic IoT environment. An AI-based detection framework is designed, incorporating supervised learning algorithms to analyse traffic patterns of the simulated environment and identify malicious behaviours in real time. A smart home environment is simulated using Home Assistant OS and virtual IoT devices which communicate via the MQTT protocol. DDoS and normal traffic are generated using python scripts, and the logs are collected to form a dataset. Characteristics such as the frequency of the message, and the device activity are extracted for the purpose of training the Isolation Forest model with the capability of identifying abnormal traffic patterns.

The outcome validates the outcome being able to detect DDoS behaviour in real time with high accuracy and reduced false positives. This solution is designed to be non-intrusive and operates outside of the smart home platform hence not modifying its core functions. The findings of this research show the potential of unsupervised machine learning in enhancing the security of IoT-based smart home.

This research contributes to the growing body of knowledge in AI-driven cybersecurity for smart homes and offers practical recommendations for making improvement to the resilience of IoT infrastructures against DDoS attacks.

Table of Contents

Declaration.....	2
Abstract	4
1. Introduction	10
1.1 Background of Study	10
1.2 Problem Statement	11
1.3 Purpose of the Study	12
1.4 Research Aim and Objectives.....	12
1.5 Research Questions and Hypotheses	13
1.6 Significance of the Study	14
1.7 Scope and Limitations	14
2. Literature Review	15
2.1 Introduction.....	15
2.2 IoT Smart Home Environments and Security Challenges	16
2.3 Understanding DDoS Attacks in IoT Networks.....	17
2.4 Limitations of Traditional Intrusion Detection Systems.....	19
2.5 Machine Learning in Intrusion Detection	20
2.5.1 Supervised Learning Approaches	21
2.5.2 Unsupervised and Anomaly Detection Techniques	21
2.5.3 Deep Learning and Hybrid Models.....	22
2.5.4 Real-Time Considerations	23
2.6 MQTT Traffic as a Detection Vector	23
2.6.1 MQTT Vulnerabilities in Smart Homes.....	24
2.6.2 Traffic Features for Anomaly Detection	24
2.6.3 Advantages of Focusing on MQTT for Detection	24
2.7 Lightweight IDS Models for IoT	25
2.7.1 The Need for Lightweight Solutions	25
2.7.2 Effective Feature Selection	26
2.7.3 Isolation Forest for Lightweight Detection	26
2.7.4 XGBoost for Lightweight Detection	26
2.7.5 Edge and Fog Computing Integration	27
2.7.6 Summary	27

2.8 Summary of the Chapter.....	27
3. Methodology	28
3.1 Introduction.....	28
3.2 Research Design	29
3.3 System Architecture	31
3.3.1 Smart Home Hub and MQTT Broker	31
3.3.2 Normal Traffic Generation.....	31
3.3.3 DDoS Traffic Simulation.....	32
3.3.4 Traffic Capture with Wireshark	32
3.3.5 Data Flow Summary	33
3.4 Data Generation and Traffic Collection	34
3.4.1 Normal Traffic Generation.....	34
3.4.2 DDoS Traffic Simulation.....	34
3.4.3 Traffic Capture Using Wireshark	35
3.4.4 Dataset Structure and Preparation	35
3.4.5 Reproducibility	36
3.5 Feature Engineering.....	36
3.5.1 Data Cleaning and Preprocessing.....	36
3.5.2 Categorical Encoding	36
3.5.3 Numerical Feature Extraction	37
3.5.4 Feature Preparation for Isolation Forest	37
3.5.5 Feature Preparation for XGBoost	37
3.6 Machine Learning Model Development	38
3.6.1 Model Selection Rationale	38
3.6.2 Dataset Input Preparation.....	38
3.6.3 Model Training	39
3.6.4 Model Evaluation	39
3.7 Smart Home Integration.....	39
3.7.1 Non-intrusive Monitoring Architecture	40
3.7.2 Traffic Capture Workflow	40
3.8 Evaluation Strategy.....	41
4. Result and Analysis.....	42

4.1 Home Assistant Simulation Results	42
4.1.1 Normal Senor Traffic Behaviour	42
4.1.2 MQTT Traffic Patterns During Simulation	43
4.1.3 Behaviour Under DDoS Flooding Simulation	44
4.1.4 Dataset Generation Outcomes	44
4.1.5 Impact on IDS Model Performance	45
4.2 Isolation Forest Results	45
4.2.1 Anomaly Score Behaviour	45
4.2.2 Detection Performance	46
4.2.3 Confusion Matrix Analysis	46
4.2.4 Overall Detection Ability	47
4.2.5 Interpretability of Results	47
4.3 XGBoost Results	47
4.3.1 Classification Metrics	48
4.3.2 Confusion Matrix Interpretation	48
4.4 Models Integration with Home Assistant	49
4.4.1 Isolation Forest Model Integration with Home Assistant	49
4.4.2 XGBoost Model Integration with Home Assistant	52
5. Conclusion and Recommendations	55
5.1 Conclusion	55
5.2 Research Contributions	56
5.3 Recommendations for Future Work	57
5.3.1 Incorporation of Real-World Traffic Data	57
5.3.2 Feature Alignment Between Training and Deployment	57
5.3.3 Hybrid and Layered Detection Architecture	57
5.3.4 Continuous Learning and Model Adaptation	57
5.3.5 Enhanced Smart Home Response Mechanisms	58
5.4 Final Remarks	58
References	58
Appendices	61
Appendix 1: GitHub Repository	61
Appendix 2: Screenshots of Codes and other Configurations	61

List of Figures

Figure 1 Schematic Diagram of DDoS Attack (Lee et al., 2022)	19
Figure 2 Supervised Learning (Baheti, 2021)	21
Figure 3 Unsupervised Machine Learning (GeeksforGeeks, 2023)	22
Figure 4 Normal Traffic Capture on Wireshark	32
Figure 5 DDoS Traffic Capture on Wireshark	33
Figure 6 Data Flow Summary.....	33
Figure 7 Traffic Capture Workflow	41
Figure 8 Sensor Results on Home Assistant	43
Figure 9 Normal Traffic Sensor simulation	61
Figure 10 Normal Traffic Sensor simulation	62
Figure 11 DDoS traffic sensor simulation	63
Figure 12 DDoS traffic sensor simulation.....	63
Figure 13 DDoS traffic sensor simulation.....	64
Figure 14 Isolation Forest Model building	64
Figure 15 Isolation Forest Model building	65
Figure 16 Isolation Forest model evaluation	65
Figure 17 Isolation Forest model evaluation.....	66
Figure 18 Isolation Forest model evaluation.....	66
Figure 19 Isolation Forest Integration using Flask API and Detection	67
Figure 20 Isolation Forest Integration using Flask API and Detection.....	68
Figure 21 Isolation Forest Integration using Flask API and Detection.....	68
Figure 22 Isolation Forest Integration using Flask API and Detection.....	69
Figure 23 Isolation Forest Integration using Flask API and Detection.....	69
Figure 24 XGBoost model building and evaluation	70
Figure 25 XGBoost model building and evaluation.....	70
Figure 26 XGBoost model building and evaluation.....	71
Figure 27 XGBoost model building and evaluation.....	71
Figure 28 XGBoost model integration with Home Assistant and live Detection	72
Figure 29 XGBoost model integration with Home Assistant and live Detection	72
Figure 30 XGBoost model integration with Home Assistant and live Detection	73
Figure 31 XGBoost model integration with Home Assistant and live Detection	73
Figure 32 XGBoost model integration with Home Assistant and live Detection	74
Figure 33 XGBoost model integration with Home Assistant and live Detection	74
Figure 34 Flask API on Powershell.....	75
Figure 35 Home Assistant Yaml Configuration	75
Figure 36 Home Assistant Yaml Configuration	76

Figure 37 Normal Traffic capture	76
Figure 38 DDoS Traffic capture	77
Figure 39 Home Assistant CLI	77
Figure 40 Home Assistant Integrations.....	78

1. Introduction

1.1 Background of Study

The fast growth of the Internet of Things (IoT) has changed modern homes into an interconnected digital ecosystem. Various type of devices such as smart thermostats, security cameras, light bulbs, and voice-controlled assistants has all become common features in residential environments. These devices provide convenience, automation, and energy efficiency. Nevertheless, they also introduce new cybersecurity vulnerabilities due to the limited computing power they possess, inconsistent security standards, and frequent exposure to the internet.

Smart home networks use protocols such as MQTT (Message Queuing Telemetry Transport) for lightweight communication between the devices and the central hub. MQTT is an efficient protocol but lacks strong built-in security features. Resulting to exploit of IoT protocols to launch Distributed Denial-of-Service (DDoS) attacks. These attacks flood a network or service with a large volume of traffic, which leads to disruption or total failure of services. A notable example is the Mirai botnet, which hijacked thousands of vulnerable IoT devices by using a table of more than 60 common factory default usernames and passwords, and logs into them to infect them (Affinito *et al.*, 2023).

Traditional Intrusion Detection Systems (IDS) are not properly suited to detect these types of threats in smart home environments. They only rely on static rules or known attack signatures, which are mostly ineffective against these evolving attacks. Additionally, smart home devices have limited resources, making it unrealistic to deploy heavyweight security solutions directly on them.

Artificial Intelligence (AI), particularly machine learning (ML), provides a capable alternative. ML-based detection systems can learn from patterns in network traffic and abnormal behavior that may imply an ongoing attack. This approach supports real-time anomaly detection and adjusts to new attack techniques without any manual updates.

This study aims to apply AI technique to the detection of DDoS attacks within a simulated smart home environment in Home Assistant using MQTT traffic. It investigates how lightweight ML models like Isolation Forest can be used to detect anomalies in traffic patterns and provides response in real time, without having any effect on the device performance. The

goal is to create an efficient and scalable detection mechanism that fits the distinctive requirements of smart homes.

The convergence of IoT, cybersecurity, and AI makes this research timely and necessary. With the increase in the adoption of smart home technologies worldwide, the need for intelligent, real-time protection against cyber threats is very critical.

1.2 Problem Statement

Smart home environments are rapidly becoming targets of cybercriminals due to the weak security posture of IoT devices. These devices mostly lack strong encryption, firmware updates, and authentication mechanisms. Their lightweight nature and always-on connectivity make them attractive entry points for large-scale Distributed Denial-of Service (DDoS) attacks. When they get compromised, they can either be the target of the attack or be used as part of a botnet to disrupt services.

Existing IDS are mainly designed for traditional IT infrastructures, not the dynamic and heterogeneous nature of smart homes. Signature-based IDS tools find it difficult to detect novel DDoS patterns, especially in MQTT-based communication environments. Additionally, deploying heavy security systems on resource-constrained IoT devices is not practical as it can degrade system performance.

There is a vivid gap in designing lightweight, real-time, and intelligent security solutions tailored to smart home environments. While AI and ML models have clearly shown capacity in anomaly detection, many existing approaches either have their focus on enterprise networks or lack integration with real-world smart home platforms.

This research addresses the following major challenge: how to apply an AI-based approach to efficiently detect DDoS attacks in smart home networks using lightweight, scalable techniques without making any modification to the core system functionalities. The focus is mainly on detecting anomalies in MQTT traffic using machine learning and integrating the detection algorithm into Home Assistant; a widely used smart home hub, without having any additional system overhead.

Without fixing this challenge, smart home users will remain exposed to severe threats that can compromise privacy, security, disrupt services, and undermine trust in IoT technologies.

1.3 Purpose of the Study

The purpose of this study is to develop and evaluate an AI-based detection system with the capability of detecting Distributed Denial-of-Service (DDoS) attacks in IoT-enabled smart home environments. The research aims to address the limitations faced in using traditional intrusion detection systems by applying machine learning techniques to analyse MQTT traffic patterns for anomalies.

This study focuses on creating a lightweight and non-intrusive model that can run with smart home platforms like the Home Assistant without influencing their core operations. The goal is to simulate both normal and attack traffic scenarios, capture the traffic using Wireshark, extract meaningful features from the data, and train a machine learning model for real time detection of DDoS attacks.

By achieving this, the study seeks to contribute to the building of more resilient and secure smart home systems. It provides practical insight into the integration of AI-driven cybersecurity solutions within existing IoT infrastructures, enhancing the protection of consumer of smart home environments against evolving cyber threats.

1.4 Research Aim and Objectives

Research Aim

The aim of this research is to design and implement an AI-based intrusion detection system that can precisely detect DDoS attacks in smart home environments by analysing MQTT traffic. The system should be lightweight, real-time, and compatible with existing smart home platforms such as the Home Assistant.

Research Objectives

To fully achieve this aim, the study sets out the following specific objectives:

1. Simulate a smart home environment using Home Assistant and virtual IoT sensors that communicate using MQTT protocol.
2. Generate both normal and DDoS traffic patterns using python scripts to create a realistic dataset for training and testing.

3. Extract and engineer relevant features from MQTT traffic, such as the frequency of the message, device activity, and payload size.
4. Train and evaluate two machine learning model, an unsupervised learning model such as the Isolation Forest machine learning model, and a supervised learning model such as the Extreme Gradient Boost (XGBoost) model to perform the detection of anomalies in the MQTT traffic that signifies a DDoS attacks.
5. Integrate the AI detection model with the smart home environment in a non-intrusive manner, making sure that it does not affect the system performance of the Home Assistant.
6. Assess the machine learning model's performance in terms of detection accuracy, false positive rate, and effectiveness of the real-time detection.
7. Compare the AI-based approach with the traditional signature-based methods of detection to figure out the advantages it has in a smart home environment.

1.5 Research Questions and Hypotheses

Research Questions

This research is guided by the following research questions:

1. How can artificial intelligence be applied to effectively detect and mitigate DDoS attacks targeting IoT devices in smart home environments?
2. What patterns in MQTT traffic indicate a DDoS attack in a smart home environment?
3. Can a lightweight, non-intrusive AI model be integrated with existing smart home platforms like Home Assistant without having a degrading effect on the system performance?

Research Hypotheses

To address the aforementioned research questions, the following hypotheses are proposed:

1. H1: AI-based models, such as Isolation Forest and XGBoost, can help detect DDoS attack patterns in MQTT traffic with an expected higher accuracy and lower false positives compared to available traditional signature-based methods.

2. H2: Simulated IoT traffic using virtual sensors and Python MQTT scripts can reliably replicate real world DDoS attack scenarios, resulting in valid datasets for training and testing detection models.

1.6 Significance of the Study

This study addresses a growing worry in cybersecurity which is the protection of smart home environments against Distributed Denial-of-Service (DDoS) attacks. As IoT devices become more widely used, so do the risks associated with their usage rises. Many of these IoT devices lack adequate security features, making them a very attractive targets for attackers. This creates a need for new approaches that can provide real-time protection against attacks without influencing the device resources.

The significance of this research lies in its practical application of artificial intelligence to solve this problem. By developing a lightweight and non-intrusive detection system, the research shows how smart homes can be properly secured using machine learning without needing to modify the core architecture of smart home existing platforms like the Home Assistant.

The project adds to ongoing discussions around AI-driven cybersecurity by showing that unsupervised machine learning techniques such as Isolation Forest can detect abnormal traffic patterns linked with DDoS attacks. It also introduces a framework for simulating smart home traffic and attacks using generally available tools, providing a replicable model for future study.

Additionally, this work provides insights for developers, researchers and cybersecurity professionals on how AI can be integrated into IoT ecosystems. It supports the development of scalable, intelligent security mechanisms appropriate for home networks, which in turn build trust in smart home technologies.

1.7 Scope and Limitations

Scope of the Study

This study focuses on the detection of Distributed Denial-of Service (DDoS) attacks in IoT-based smart home environments using artificial intelligence (AI). The core elements within the scope includes:

- Simulating a smart home environment using Home Assistant deployed via a virtual machine.
- Generating normal and DDoS traffic over MQTT protocol using Python scripts.
- Capturing the normal and DDoS MQTT traffic data using Wireshark.
- Ensuring the AI model is lightweight and non-intrusive, fit for real-time use alongside smart home platforms.
- Evaluating the model's performance based on accuracy, false positives, and the speed of detection.

The study uses virtual devices/sensors and software-based simulations to replicate real-world situations in a controlled lab environment.

Limitations of the Study

Regardless of its contributions, the study has various limitations:

- Simulated environment: The smart home environment setup is virtual and may not fully depict the complexity or variability of real-world IoT deployments.
- Focus on MQTT protocol: Although MQTT is widely used, the study does not account for other IoT communication protocols such as CoAP or Zigbee.
- Detection only, not prevention: The project focuses on detecting attacks, not on automating a response to prevent or defence mechanism.
- Limited size of dataset: Data used for training and testing are generated through simulation, which may limit generalisation to large scale, real-world scenarios.

These limitations help provide insights to the research's boundaries, hence, identifying opportunities for future work.

2. Literature Review

2.1 Introduction

The increasing rate of the adoption of Internet of Things (IoT) devices in smart home environments has further raised concerns about network security, especially in the face of Distributed Denial-of-Service (DDoS) attacks. These attacks take advantage of the weak

security features of IoT devices and protocols, targeting centralised systems like the home automation hubs and MQTT brokers. As homes become more smart and more connected to the internet, traditional security frameworks that are designed for enterprise networks often find it difficult to keep up with the scale, diversity, and dynamism of consumer IoT ecosystems.

This chapter focuses on the review of key studies and emerging technologies that are relevant to AI-based intrusion detection in smart homes. The literature review starts with an overview of smart home architecture and its inherent security vulnerabilities. It then explores the features of DDoS attacks in IoT networks and evaluates the limitations of traditional intrusion detection systems (IDS). The chapter further proceeds to explore the application of artificial intelligence, especially machine learning, in the development of adaptive and lightweight detection systems that are suitable for resource-constrained environments. Focus is on the MQTT which is the primary protocol for communication in a lot of smart home setups and how the traffic patterns generated from the communication can be used in detection of malicious activity.

The goal of this review is to establish a solid theoretical ground for the development of AI-based anomaly detection model designed to smart home environments. It emphasizes the gaps in existing research, justifies the selection of the machine learning model used (Isolation Forest), and identifies opportunities for designing practical, scalable security solutions.

2.2 IoT Smart Home Environments and Security Challenges

The concept of a smart home centers around the integration of various IoT devices to provide automation, control and monitoring capabilities. These devices vary from smart bulbs, thermostats, locks, and cameras to voice assistants and home hubs. Majority of these devices communicate via wireless protocols and are centrally managed using platforms like the Home Assistant or SmartThings. The goal is to ensure a seamless and intelligent living environment is created, however, this interconnectivity usher in serious security vulnerabilities.

Smart home devices often use lightweight communication protocols like the MQTT, Zigbee, Bluetooth Low Energy (BLE), or Wi-Fi. While MQTT is preferred because of its low bandwidth requirements and publish-subscribe architecture, it does not have a strong native encryption and access control mechanisms (Ahmed and Zeebaree, 2021). This makes it highly vulnerable to traffic manipulation, replay attacks, and message flooding, especially when it is used with weak or default configurations.

Many IoT devices are developed with minimal emphasis on security. Manufacturers often lay more priority on the cost, battery efficiency, and fast deployment over secure software development practices. Which often results in devices ship with hardcoded credentials are lacking firmware update mechanisms, or rely on insecure APIs (Darem, Alhashmi and Jemal, 2022). These limitations make them captivating targets for cybercriminals who exploit them to launch large scale attacks, as seen with the Mirai botnet and its many variants.

The distributed nature of smart homes makes security more complicated. Unlike centralised enterprise networks, smart homes consist of heterogeneous devices from different manufacturers, each with its respective firmware, communication standard, and update lifecycle. This further makes the implementation of a unified security policy or detection mechanism difficult (Garba *et al.*, 2024). Additionally, most homeowners lack the technical knowledge to properly configure secure networks or recognize attack symptoms in real time.

In addition, smart home devices mostly run continuously and remain connected to the internet, which further increases their exposure to cyber threats. Because the IoT devices have limited processing power, memory, and battery life, deploying conventional Intrusion Detection System (IDS) directly on them is mostly impractical (Sadhwani *et al.*, 2023). This has then shifted attention to external network-based detection methods that monitor traffic centrally and implement AI models to identify abnormal behavior.

In summary, the expansion of smart home technologies has introduced multiple entry points for cyber-attacks. Their limited security capabilities, dependence on insecure protocols, and lack of user awareness create a crucial need for intelligent, lightweight, and protocol-aware security mechanisms tailored particularly for IoT environments.

2.3 Understanding DDoS Attacks in IoT Networks

Distributed Denial-of-Service (DDoS) attacks are one of the most regularly occurring and disruptive form of cyberattacks in IoT environments. A DDoS attack overwhelms a target such as a server, device, or network with high volumes of malicious traffic beyond what it has the capability of handling from multiple sources. In IoT-based smart homes, attackers mostly hijack vulnerable devices and turn them into part of a botnet, which are then used to flood MQTT brokers, routers, or cloud services with traffic beyond its capability (Alashhab *et al.*, 2024).

The nature of IoT makes these types of attacks very dangerous and difficult to detect. Most devices do not have proper access control and run outdated firmware, giving room to attackers to easily gain unauthorised access. Once the devices get compromised, they can send rapid, repetitive messages to other nodes or central brokers, draining bandwidth, exhausting CPU cycles, or leading to application crashes (Sarwar *et al.*, 2023).

Smart homes are particularly vulnerable due to their reliance on lightweight protocols like the MQTT, which when being designed, security was not a factor. Attackers exploit this by sending massive volumes of publish messages with high frequency, leading to application-layer DDoS conditions. MQTT brokers such as Mosquitto may crash or become unresponsive when it gets flooded with excessive client connections or topic updates (Alahmadi *et al.*, 2023)

Historically, the Mirai botnet showed the destructive potential of DDoS attacks launched through IoT devices that have been hijacked. It targeted poorly secured cameras, Digital Video Recorders (DVRs), and routers, leading to outages across major websites and services (Alahmadi *et al.*, 2023). Ever since, DDoS toolkits have evolved, becoming more sophisticated and difficult to trace. Attackers are now using randomized payloads, dynamic IP sources, and encrypted traffic to avoid being detected.

Recent reports have shown a surge in IoT-targeted DDoS attacks. According to Nokia's Threat Intelligence Report, IoT devices accounted for over 32% of all infected devices used in DDoS botnets in 2023, with an increase rate of 500% in new variant of botnets observed in just a year (Darem, Alhashmi and Jemal, 2022). These statistics underline the growing urgency to detect and respond to DDoS attempts in real time.

Furthermore, the impact of a DDoS attack extends beyond denial of service. It can reduce the performance of critical systems such as smart locks, alarms and Heating, Ventilation and Air Conditioning (HVAC) systems, which is a major threat to physical safety. In some cases, sustained attacks can serve as a distraction while other intrusions such as data theft are being carried out in parallel (Nawaal *et al.*, 2024).

Given the unpredictable and evolving nature of DDoS threats in smart homes, there is a need for adaptive detection mechanisms that monitor behavioural patterns in traffic rather than solely relying on attacks signatures that are previously known. This sets the stage for the introduction of AI and machine learning as dynamic tools for the identification of such anomalies efficiently.

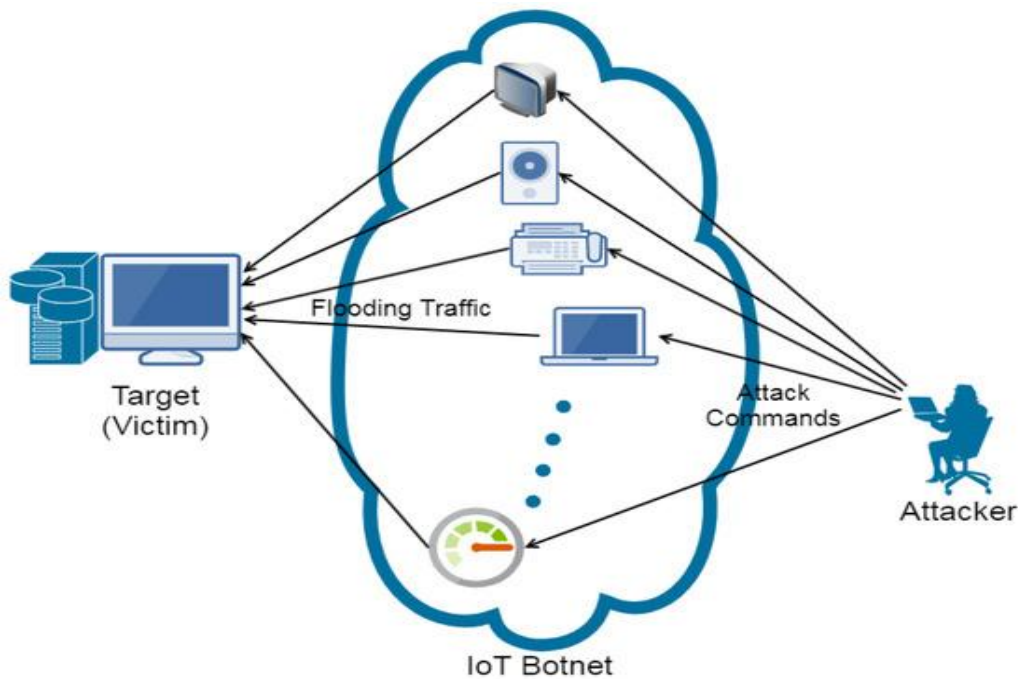


Figure 1 Schematic Diagram of DDoS Attack (Lee et al., 2022)

2.4 Limitations of Traditional Intrusion Detection Systems

Intrusion Detection Systems (IDS) are generally used in enterprise environments to detect and respond to malicious activity on the network. These two systems are mainly classified into two categories: signature-based IDS and anomaly-based IDS. Signature-based systems depend on those predefined patterns or attack signatures to identify threats, while anomaly-based systems identify deviations from established baseline behaviour. However, the two approaches face serious challenges when implemented in an IoT-based smart home environments.

One of the major limitations of traditional IDS in smart homes is the inefficiency against threats that were not previously known. Signature-based detection mechanisms can only help with the detection of attacks for which the signatures already exist. This makes them inefficient against zero-day exploits or novel variations of existing attacks (Nawaal *et al.*, 2024). In the perspective of DDoS attacks, which are always rapidly evolving, signature-based systems tend to fail to recognize subtle changes in traffic behaviour.

Another notable issue is the high resource demand of traditional IDS. Many commercial and open-source systems are made for high-performance computing environments and need considerable processing power, memory, and storage which are lacking in IoT devices. IoT devices in smart homes, by contrast, are mostly built with limited computational capabilities,

making it impractical to deploy conventional IDS directly on these devices (Sadhvani *et al.*, 2023).

Additionally, scalability often becomes a concern in dynamic and heterogeneous IoT environments. Smart homes can contain numerous numbers of devices from multiple vendors, each using different protocols for communication and data formats. This diversity complicates the implementation of unified policies for detection and increases the possibility of false positives and miscalculations (Indira and Sakthi, 2020).

Furthermore, lack of context awareness reduces the effectiveness of traditional IDS. These systems mainly analyze traffic in isolation and do not take into consideration the behaviour of the device, intention of the user, or environment context. As a result, normal but unusual events such as when a user is streaming video while downloading updates, this may be flagged as suspicious, while subtle indicators of malicious activity may be overlooked (Ahmed and Zeebaree, 2021).

In conclusion, traditional IDS systems mostly lack real-time responsiveness. Many rely on periodic analysis of log or batch processing rather than real-time monitoring and alerting. In a smart home environment, where critical devices such as smart locks or alarms may be affected during a case of an attack, delayed detection and response can have serious (Alahmadi *et al.*, 2023).

These listed limitations highlight the need for smarter, more adaptive, and lightweight detection mechanisms. AI-based systems, especially those that utilizes machine learning for behaviour analysis, offer a more promising alternative by learning traffic patterns and detecting anomalies in real time with minimal resource overhead.

2.5 Machine Learning in Intrusion Detection

The integration of machine learning (ML) into intrusion detection systems has received an increase in attention as a solution to the challenges faced while using traditional methods, particularly in dynamic and resource-constrained environments such as the IoT-based smart homes. Unlike signature-based systems, machine learning models can generalize from data, making them to detect novel or evolving threats without any prior knowledge of attack signatures.

2.5.1 Supervised Learning Approaches

Supervised learning models are mainly used in network security to classify traffic as benign or malicious based on labelled datasets. Algorithms such as XGBoost, Support Vector Machines (SVM), Random Forest, and Decision Tree have all shown their strong performance in detecting various types of attacks, including DDoS (Indira and Sakthi, 2020). These models all work by learning from features such as size of the packet, frequency, source IP, and type of protocol.

Nevertheless, supervised models face two major limitations in smart home contexts. First, labelled datasets are quite scarce and a bit difficult to obtain, especially for emerging attack types or protocol specific such as MQTT. Second, the high diversity of IoT devices and traffic patterns may cause reduction in the generalizability of models trained on static datasets (Nawaal *et al.*, 2024)

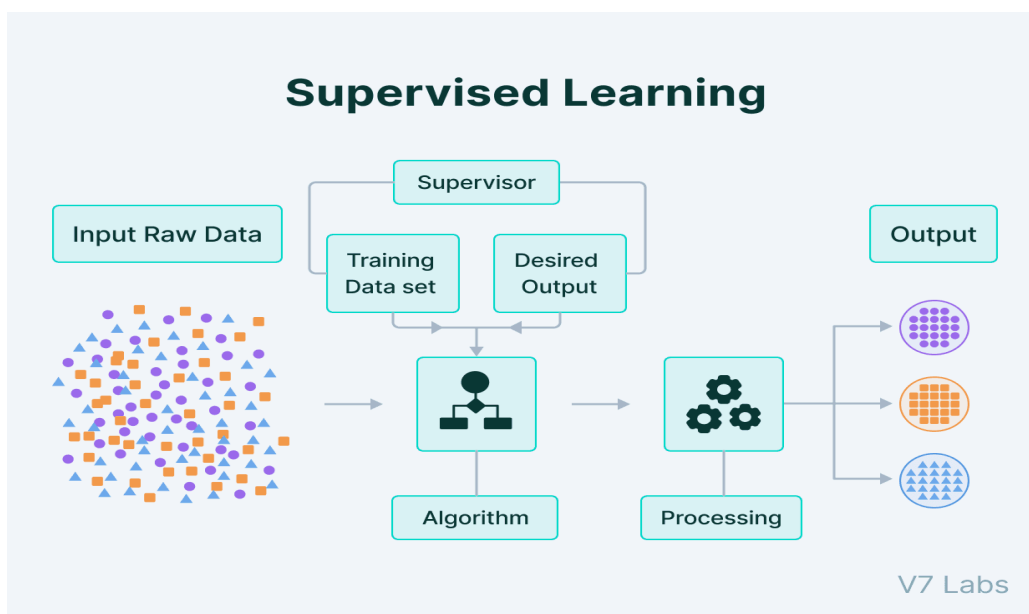


Figure 2 Supervised Learning (Baheti, 2021)

2.5.2 Unsupervised and Anomaly Detection Techniques

To conquer the limitations of supervised methods, researchers have explored the option of unsupervised learning techniques that do not require labelled data. These models detect deviation from normal behaviour and flag them as potential intrusions. Isolation Forest, K-

means clustering, and One-Class SVM are among the most widely used algorithms in this category of unsupervised learning (Ram *et al.*, 2025).

Taking particular attention to one of the algorithms used in unsupervised learning, which is the Isolation Forest, it is very well suited for smart home environments due to its ability to isolate anomalies effectively in high-dimensional environments with low-computational cost. It builds random trees and evaluates how promptly a data point gets isolated, assuming that anomalies are easier to isolate than normal observations (Alahmadi *et al.*, 2023).

Unsupervised models are particularly useful for detecting zero-day or previously unknown attacks. They adapt to traffic that changes pattern and behaviour of the device, making them more flexible than static rule-based systems (Sadhvani *et al.*, 2023).

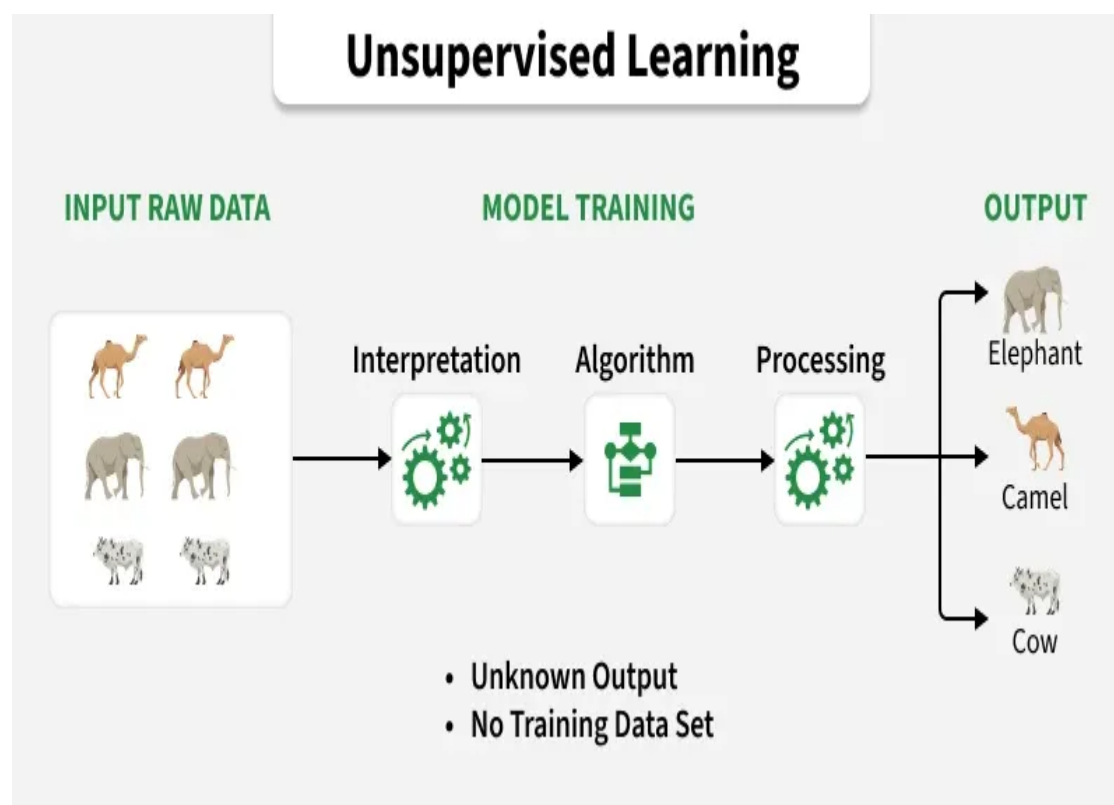


Figure 3 Unsupervised Machine Learning (GeeksforGeeks, 2023)

2.5.3 Deep Learning and Hybrid Models

Recent research has also explored deep learning techniques such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) networks for the purpose of intrusion detection. These models have the capability to extract

complex patterns from raw traffic data and have shown much higher accuracy in controlled environments (Hussain *et al.*, 2020).

Hybrid models that combine machine learning with heuristic rules or ensemble learning methods have been proposed to better improve the rate of detection and reduce false positives. For instance, integrating an Isolation Forest with an LSTM model can provide both temporal behaviour analysis and statistical anomaly detection, increasing overall robustness (Iliyasu *et al.*, 2025).

Nevertheless, deep learning models mostly require significant computational resources, which hinders their deployment on low-power IoT devices. For this reason, lightweight AI approaches that ensures accuracy and efficiency remain a more go to solution in smart home security.

2.5.4 Real-Time Considerations

A major benefit of Machine Learning based IDS is the potential for detection in real time. Many models can be integrated with message brokers or edge devices to analyse traffic during process or operations executions. For example, MQTT brokers can forward message statistics to an ML engine that signals anomalous behaviour without causing an interruption to the device communication (Chen *et al.*, 2020).

Real-time ML detection is especially important in smart homes, where delays in detecting threats can cause disruption to critical services such as surveillance or environmental control systems.

2.6 MQTT Traffic as a Detection Vector

Message Queuing Telemetry Transport (MQTT) is one of the most widely accepted communication protocols used in smart home environments due to its simplicity, efficiency and low bandwidth usage. It follows a publish-subscribe model, where IoT devices (publishers) send messages to a central broker, which then forwards the messages to devices (subscribers) interested in specific topics. This architecture reduces overhead but introduces a range of vulnerabilities that make it a potential target for Distributed Denial-of-Service (DDoS) attacks.

2.6.1 MQTT Vulnerabilities in Smart Homes

Despite its numerous advantages, MQTT was not designed with strong security characteristics. It offers basic authentication and optional Secure Sockets Layer/Transport Layers Security (SSL/TLS) encryption, which are mostly not implemented due to resource constraints or oversight by the developer (Aysa, Ibrahim and Mohammed, 2020). As a result, attackers can then exploit insecure MQTT brokers by sending rapid publish requests or distorted payloads that disrupt communication.

In many smart home setups, MQTT brokers such as Mosquitto are deployed without rate-limiting or robust authentication. This gives room to application-layer DDoS attacks, where broker becomes overwhelmed by frequent messages from many sources. These attacks slow down the broker speed and also can cause it to totally crash, leading to loss of communication across all connected devices (Sarwar *et al.*, 2023).

2.6.2 Traffic Features for Anomaly Detection

Studies shows that specific patterns in MQTT traffic can indicate malicious behaviour. These include:

- Uncommonly high frequency of message per device.
- Topic subscriptions or rapid switching of topic which are not regular.
- Large size of payloads or repeated identical messages.
- Unexpected spikes in connection and disconnection events.

Monitoring these characteristics allows for effective detection of anomaly detection using machine learning. For instance, Isolation Forest models can be trained to identify when a device deviates notably from its normal traffic pattern, signalling a possible DDoS attack (Ram *et al.*, 2025).

(Alahmadi *et al.*, 2023) used MQTT message rates and timing intervals to train an unsupervised anomaly detection model, accomplishing high accuracy in detecting attacks without needing deep inspection of packet. This makes MQTT traffic a rich, low-cost source of data for real-time monitoring and detection of intrusion.

2.6.3 Advantages of Focusing on MQTT for Detection

Focusing on MQTT traffic provides several benefits for security in smart homes:

- **Lightweight analysis:** Traffic metadata (e.g., message count, topic structure) can be gathered without an active inspection of message content, preserving privacy and reducing overhead.
- **Protocol centrality:** Since MQTT mostly acts as the main channel of communication, securing it will strengthen the overall resilience of the network.
- **Ease of integration:** Most brokers support logging, application programming interface (API) access, or command line interface (CLI) tools, which makes integration with external AI models simplified.

In addition, MQTT brokers can be configured to export logs to external scripts or services, giving room to the operation of detection models independently of the smart home platform's core logic, making it perfect for non-intrusive deployments.

2.7 Lightweight IDS Models for IoT

Intrusion Detection Systems (IDS) for traditional IT infrastructure are mostly resource-intensive and not properly suited for IoT environments, where devices operate under strict hardware and energy constraints. In smart homes, deploying IDS that are not only effective but also lightweight is very critical to making sure that real-time protection without compromising the performance of the device or user experience.

2.7.1 The Need for Lightweight Solutions

Smart homes IoT devices generally possess limited processing power, memory, and battery life. Conventional IDS, especially those based on deep inspection of packet or complex extraction of feature, can easily overwhelm these systems (Sadhvani *et al.*, 2023). Hence, there is a need for models that can perform traffic analysis effectively using minimal computational resources.

Lightweight IDS must strike a balance between accuracy, speed, and system overhead. To achieve this, a lot of recent research has turned to statistical and machine learning-based anomaly detection techniques that rely on a small number of simple traffic features (Ma *et al.*, 2025).

2.7.2 Effective Feature Selection

Feature engineering is a core component of lightweight IDS. The goal is to identify a minimal set of indicators that are informative enough to detect behaviour that are abnormal while needing little computational effort to process.

Common features used in lightweight models include:

- Frequency of message per device per second.
- Diversity of topic or entropy.
- Average payload size.
- Connection and disconnection rates.
- Packet inter-arrival times.

By paying attention on these features, models can efficiently detect patterns associated with DDoS attacks without performing any analysis on the payload content or complex protocol headers (Darem, Alhashmi and Jemal, 2022).

2.7.3 Isolation Forest for Lightweight Detection

Isolation Forest has appeared as a popular choice for lightweight anomaly detection in IoT due to its low computational complexity and high accuracy rate. The model works by randomly partitioning data and measuring how instances can be isolated easily, anomalies are isolated faster than normal instances (Ram *et al.*, 2025).

Its simplicity allows for fast training and prediction, making it a perfect fit for deploying on edge gateways or low-power devices. (Ram *et al.*, 2025) showed that Isolation Forest detected MQTT-based DDoS attacks with an accuracy of over 95% and minimal false positives in simulated smart home traffic.

2.7.4 XGBoost for Lightweight Detection

Extreme Gradient Boost (XGBoost) is a high-performance ensemble learning algorithm widely used for lightweight detections in cybersecurity. Its design focuses on speed, scalability, and efficiency, which makes it a good fit for resource-constrained environments such as IoT devices and edge computing systems. XGBoost's parallelized tree construction, regularization techniques, and sparse-aware algorithms makes it capable of handling large datasets with minimal computational overhead. When it comes to intrusion detection systems (IDS),

XGBoost has shown a greater level of accuracy and responsiveness compared to other traditional classifiers, especially in terms of identifying threats such as DDoS attacks, port scans and malicious activity (Sah *et al.*, 2024)

2.7.5 Edge and Fog Computing Integration

Recent body of work has proposed the deployment of lightweight IDS at the edge or fog layer rather than on individual IoT devices. This gives the opportunity for a centralised monitoring of multiple devices while still maintaining low latency and high responsiveness (Guezzaz *et al.*, 2022). Edge-based systems can perform collection and processing of traffic locally, reducing reliance on cloud infrastructure and improving accuracy.

Hybrid frameworks have also been developed, which is done by combining lightweight models with rule-based systems to further improve detection coverage without increasing the demands on resources (Alahmadi *et al.*, 2023).

2.7.6 Summary

Lightweight IDS are very important for practical applications of smart home security. Models such as the Isolation Forest provide a strong trade-off between efficiency and accuracy, particularly when deployed with proper selection of feature and edge-based integration. Future research should pay more attention on optimising these models for real-world smart home platforms and validating the models across various IoT setups.

2.8 Summary of the Chapter

This chapter has explored the main areas of research related to AI-based detection of DDoS attacks in IoT-enabled smart home environments. It started by firstly examining the architecture and communication protocols used in smart homes, highlighting the widespread dependence on MQTT and the security vulnerabilities that are associated with the communication protocol. The limited processing capabilities of IoT devices, alongside its weak authentication mechanisms, makes smart homes quite exposed to targeted cyberattacks, especially Distributed Denial-of-Service (DDoS) attacks (Alahmadi *et al.*, 2023)

The chapter further discussed the developing prevalence of DDoS attacks in IoT networks, laying emphasis on how attackers exploit unsecured devices to launch traffic floods on a large scale against MQTT brokers and smart home controllers (Alashhab *et al.*, 2024). Signature-

based Intrusion Detection Systems (IDS), while widely used in enterprise networks, they were found to be not suited for smart home contexts due to their high resource demands and inability to detect zero-day threats (Sadhvani *et al.*, 2023)

In response to these limitations mentioned, recent studies have shifted toward implementing machine learning (ML)-based IDS, especially anomaly detection models that are capable of recognising deviations from normal behaviour without the need for labelled datasets. Unsupervised algorithms such as Isolation Forest have proven to be effective in detecting DDoS anomalies in MQTT traffic, providing a strong performance with low computational overhead (Ram *et al.*, 2025). These models are particularly properly suited for smart homes where the behaviour of the traffic is relatively consistent and can be profiled effectively.

A focus on MQTT traffic as a detection vector provide practical advantages, such as lightweight data collection, ease of integration with brokers, and non-intrusive monitoring (Aysa, Ibrahim and Mohammed, 2020). In addition, lightweight IDS models that depend on simple features such as message rate and topic diversity can be implemented at the edge or fog layer to ensure both efficiency and scalability (Ma *et al.*, 2025)

In conclusion, the literature supports the feasibility and necessity of adopting AI-powered, lightweight IDS for securing smart homes. Nevertheless, a notable gap remains in the integration of such models into real-world platforms like Home Assistant. The next chapter will outline the methodology used in this study to develop and evaluate a machine learning-based detection system tailored for smart home environments.

3. Methodology

3.1 Introduction

This chapter outlines the methodology used in designing and evaluating an AI-based intrusion detection system (IDS) for the detection of Distributed Denial-of-Service attacks in an IoT-based smart home environments. This research implements an experimental design that merges simulation, data collection, and machine learning model development to address the unique challenges of smart homes.

A simulated smart home environment was implemented using Home Assistant OS as the central hub and Mosquitto MQTT broker for device communication. Virtual IoT sensors (temperature,

humidity, and gas) were setup to generate normal traffic, and Python scripts were used in simulating DDoS flooding attacks. Network traffic from both of the aforementioned scenarios was captured using Wireshark, which allowed for the extraction of structured datasets containing timestamps, source ip address, destination ip address, and packet info. Wireshark has been generally used in intrusion detection research for the generation of reliable datasets due to its packet-level precision (Darem, Alhashmi and Jemal, 2022)

Two machine learning algorithms were employed to detect anomalies in the collected traffic which are isolation Forest and XGBoost. Isolation Forest is an unsupervised anomaly detection algorithm that isolates abnormal traffic patterns based on the assumption that it is easier to separate anomalies than normal data points. It has been proven very effective in the detection of IoT-based DDoS attacks with low computational overhead (Ram *et al.*, 2025). XGBoost, on the other hand is a supervised classification algorithm which is a gradient boosting algorithm which builds decision trees sequentially, optimizing for accuracy and speed (Sah *et al.*, 2024). It has demonstrated high accuracy in the classification of IoT traffic and is particularly effective when datasets contain labelled attack and normal instances (Indira and Sakthi, 2020).

By having the combination of these two approaches, this study provides a comparative evaluation of unsupervised anomaly detection and supervised classification within the context of IoT smart homes. The application of both models allows the study to assess trade-offs between real-time adaptability and classification accuracy, which are vital in resource-constrained environments (Sadhvani *et al.*, 2023).

The methodology is structured into various stages: research design, system architecture, data generation and collection, feature engineering, model development, smart home integration, and evaluation. Each of these stages is designed to ensure that the proposed IDS remains lightweight, non-intrusive, and scalable, making it suitable for the deployment of the IDS in real-world smart home environments.

3.2 Research Design

This study follows an experimental research design, which is suitable for assessing the effectiveness of AI-based intrusion detection systems in IoT smart home environments. Experimental design permits the controlled simulation of both the normal and malicious network traffic, providing a reliable foundation for training and testing the detection models (Darem, Alhashmi and Jemal, 2022).

This study follows a quantitative method where numeric traffic data are collected, processed, and analysed with the use of statistical and machine learning methods. This is to ensure objectivity in evaluating model performance through metrics such as accuracy, precision, recall, and F1-score (Sadhvani *et al.*, 2023).

This design is structured around these three main components:

1. Simulation of IoT Traffic

- Normal traffic is generated by the configuration of virtual IoT sensors (temperature, humidity, and gas) in Home Assistant OS to publish data over a given period of time via the Mosquitto MQTT broker.
- Malicious traffic is simulated using Python-based flooding scripts that publishes data at extremely high rates, replicating an MQTT-based DDoS attack scenario.
- Wireshark is then used to capture traffic at the broker level, providing dataset containing key features such as timestamps, source ip address, destination ip address, and packet info.

2. Machine Learning Model Development

- This study uses two complementary AI approaches:
 - Isolation Forest: This is an unsupervised anomaly detection algorithm that detects irregular traffic patterns without needing labelled datasets. It is effective in detecting a zero-day or unknown attack type (Ram *et al.*, 2025).
 - Extreme Gradient Boost (XGBoost): This is a supervised classification algorithm that makes use of labelled datasets to classify traffic as benign or malicious. It has proven strong performance in the study of IoT security especially when the training data includes diverse attack scenarios (Sah *et al.*, 2024).
- By using both models, it allows for a comparative evaluation of unsupervised versus supervised detection techniques in smart home environments.

3. Evaluation Framework

- The models are being evaluated based on the following metrics: detection accuracy, false positive rate, and resource efficiency.

- Comparative testing helps to establish if a lightweight anomaly-based model like Isolation Forest can achieve similar performance levels as a more data-intensive supervised models such as the XGBoost.
- This dual evaluation helps in supporting the research aim of developing a detection system that is both effective and practical for the deployment in real-world smart home environments.

The design ensures that the methodology does not only test the feasibility of AI-based DDoS detection in IoT networks but also states trade-offs between computational efficiency and detection accuracy. Such insights appear very critical for the development of scalable and non-intrusive Intrusion Detection models for smart home environments (Alahmadi *et al.*, 2023).

3.3 System Architecture

The system architecture for this study was developed to simulate a real-world smart home environment in which both normal IoT traffic and DDoS attack traffic was generated, captured, and exported into structured CSV datasets. The setup was integrated by using Home Assistant OS, the Mosquitto MQTT broker, three virtual IoT sensors (temperature, humidity, and gas), DDoS flooding scripts using Python, and Wireshark for packet capture. This architecture made available a reproducible and controlled environment for evaluating AI-based intrusion detection models.

3.3.1 Smart Home Hub and MQTT Broker

Home Assistant OS was deployed as the central smart home hub to coordinate IoT device activity, while the Mosquitto MQTT broker was used in managing communications between devices through a publish-subscribe model. Yaml configuration can be found in Appendix 2. MQTT was chosen because of its efficiency and lightweight operation, which make it a very common protocol used in IoT smart homes. However, its lack of built-in security and rate-limiting makes it quite vulnerable to DDoS attacks that are flooding-based (Hussain *et al.*, 2020).

3.3.2 Normal Traffic Generation

Three virtual IoT sensors: temperature, humidity and gas, were configured to periodically publish readings to MQTT topics at regular intervals. This traffic generated represents benign smart home communication, where the devices send updates of environmental states to the broker without overloading it. The resulting data stream was captured and stored to represent

a Normal Traffic CSV dataset, which in it contained attributes such as timestamps, source IP, destination IP, protocol, length and info.

3.3.3 DDoS Traffic Simulation

In order to simulate a DDoS attack, custom Python flooding scripts were executed against the MQTT broker. These scripts were rapidly publishing repetitive messages to the broker at a rate much higher than the normal sensor activity, overwhelming its resources and causing degradation to the communication services. The resulting malicious traffic was then captured and saved as a DDoS Traffic CSV dataset, which displayed clear anomalies like unusually high message rates, and irregular packet timing (Darem, Alhashmi and Jemal, 2022).

3.3.4 Traffic Capture with Wireshark

All the traffics used both normal and malicious was captured at the broker level using Wireshark, a packet analysis tool that is used for recording MQTT communications with a precise high accuracy. The captured data was exported into structured CSV formats for further processing. The use of Wireshark ensured that both the normal traffic patterns and attack characteristics were reliably captured for performing feature engineering and model training (Ahmed and Zeebaree, 2021).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.61	192.168.0.15	TCP	66	53007 → 8123 [PSH, ACK] Seq=1 Ack=1 Win=253 Len=12
2	0.000009	192.168.0.61	192.168.0.15	TCP	66	[TCP Retransmission] 53007 → 8123 [PSH, ACK] Seq=1 Ack=1 Win=253 Len=12
3	0.001173	192.168.0.15	192.168.0.61	TCP	175	8123 → 53007 [PSH, ACK] Seq=1 Ack=13 Win=501 Len=121
4	0.000006	192.168.0.15	192.168.0.61	TCP	175	[TCP Retransmission] 8123 → 53007 [PSH, ACK] Seq=1 Ack=13 Win=501 Len=121
5	0.050388	192.168.0.61	192.168.0.15	TCP	54	53007 → 8123 [ACK] Seq=13 Ack=122 Win=253 Len=0
6	0.000027	192.168.0.61	192.168.0.15	TCP	54	[TCP Dup ACK 5#1] 53007 → 8123 [ACK] Seq=13 Ack=122 Win=253 Len=0
7	0.712252	192.168.0.98	192.168.0.15	UDP	220	64041 → 38900 Len=178
8	4.323147	192.168.0.15	192.168.0.61	TCP	232	8123 → 53007 [PSH, ACK] Seq=122 Ack=13 Win=501 Len=178
9	0.000016	192.168.0.15	192.168.0.61	TCP	232	[TCP Retransmission] 8123 → 53007 [PSH, ACK] Seq=122 Ack=13 Win=501 Len=178
10	0.049766	192.168.0.61	192.168.0.15	TCP	54	53007 → 8123 [ACK] Seq=13 Ack=300 Win=252 Len=0
11	0.000025	192.168.0.61	192.168.0.15	TCP	54	[TCP Dup ACK 10#1] 53007 → 8123 [ACK] Seq=13 Ack=300 Win=252 Len=0
12	0.375715	192.168.0.98	192.168.0.15	UDP	220	64041 → 38900 Len=178
13	1.746206	192.168.0.61	192.168.0.15	MQTT	85	Publish Message [home/sensor/temperature]
14	0.000017	192.168.0.61	192.168.0.15	TCP	85	[TCP Retransmission] 24760 → 1883 [PSH, ACK] Seq=1 Ack=1 Win=255 Len=31
15	0.000587	192.168.0.15	192.168.0.61	TCP	60	1883 → 24760 [ACK] Seq=1 Ack=33 Win=501 Len=0

Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device		0000 04 6c 59 d9 0e 95 04 6c 59 d9 0e 95 08 00 45 00		1y...
Ethernet II, Src: Intel_d9:0e:95 (04:6c:59:d9:0e:95), Dst: Intel_d9:0e:95 (04:6c:59:d9		0010 00 34 b7 2a 40 00 80 06 00 00 c0 a8 00 3d c0 a8		4 *@
Internet Protocol Version 4, Src: 192.168.0.61, Dst: 192.168.0.15		0020 00 0f cf 0f 1f bb 6c d6 c4 37 31 96 d1 ef 50 18	
Transmission Control Protocol, Src Port: 53007, Dst Port: 8123, Seq: 1, Ack: 1, Len: 1		0030 00 fd 81 c3 00 00 c1 86 79 50 5b c9 bb 7e 35 95	
Data (12 bytes)		0040 72 50		rP

Figure 4 Normal Traffic Capture on Wireshark

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.15	192.168.0.61	TCP	60	1883 → 42675 [ACK] Seq=1 Ack=1 Win=502 Len=0
2	0.000007	192.168.0.15	192.168.0.61	TCP	60	[TCP Dup ACK 1#1] 1883 → 42675 [ACK] Seq=1 Ack=1 Win=502 Len=0
3	0.000804	192.168.0.61	192.168.0.15	MQTT	85	Publish Message [home/sensor/temperature]
4	0.000811	192.168.0.61	192.168.0.15	TCP	85	[TCP Retransmission] 42704 → 1883 [PSH, ACK] Seq=1 Ack=1 Win=255 Len=31
5	0.000522	192.168.0.15	192.168.0.61	TCP	60	1883 → 42704 [ACK] Seq=1 Ack=32 Win=502 Len=0
6	0.000801	192.168.0.61	192.168.0.15	MQTT	85	Publish Message [home/sensor/temperature]
7	0.000805	192.168.0.15	192.168.0.61	TCP	60	[TCP Dup ACK 5#1] 1883 → 42704 [ACK] Seq=1 Ack=32 Win=502 Len=0
8	0.000803	192.168.0.61	192.168.0.15	TCP	85	[TCP Retransmission] 42693 → 1883 [PSH, ACK] Seq=1 Ack=1 Win=255 Len=31
9	0.000828	192.168.0.61	192.168.0.15	MQTT	104	Publish Message [home/sensor/humidity], Publish Message [home/sensor/gas]
10	0.000804	192.168.0.61	192.168.0.15	TCP	104	[TCP Retransmission] 42704 → 1883 [PSH, ACK] Seq=32 Ack=1 Win=255 Len=50
11	0.000418	192.168.0.15	192.168.0.61	TCP	60	1883 → 42693 [ACK] Seq=1 Ack=32 Win=502 Len=0
12	0.000805	192.168.0.15	192.168.0.61	TCP	60	[TCP Dup ACK 11#1] 1883 → 42693 [ACK] Seq=1 Ack=32 Win=502 Len=0
13	0.000836	192.168.0.61	192.168.0.15	MQTT	104	Publish Message [home/sensor/humidity], Publish Message [home/sensor/gas]
14	0.000804	192.168.0.61	192.168.0.15	TCP	104	[TCP Retransmission] 42693 → 1883 [PSH, ACK] Seq=32 Ack=1 Win=255 Len=50
15	0.000803	192.168.0.61	192.168.0.15	MQTT	85	Publish Message [home/sensor/temperature]

Figure 5 DDoS Traffic Capture on Wireshark

3.3.5 Data Flow Summary

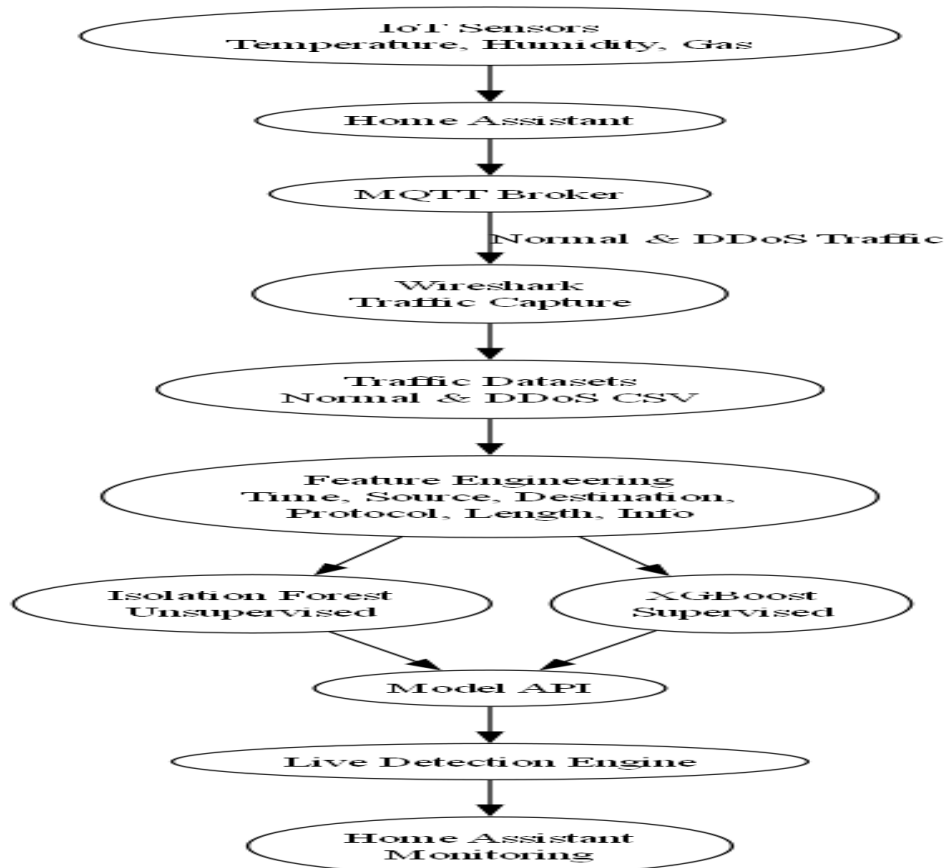


Figure 6 Data Flow Summary

This architecture used ensured that the datasets that were collected represented both benign IoT behavior and malicious DDoS conditions, making them suitable for building and evaluating lightweight AI-based intrusion detection systems.

3.4 Data Generation and Traffic Collection

This section gives an insight into how the normal IoT traffic and DDoS attack traffic were generated, captured and exported into structured CSV datasets. The final datasets used in this study were built from Wireshark captures and include the following attributes: No., Time, Source, Destination, Protocol, Length, Info. These attributes form the basis for feature engineering and AI model development.

3.4.1 Normal Traffic Generation

Normal traffic was produced by configuring three virtual IoT sensors: temperature, humidity and gas within the Home Assistant OS environment. Each sensor published readings to the Mosquitto MQTT broker at fixed intervals, ensuring the generation of stable communication patterns typical of real smart home systems. Wireshark captures this normal behaviour and exports it into normal traffic.csv, where each packet is represented using:

- No. – Packet index
- Time – Timestamp from capture start
- Source – Device or broker IP sending the packet
- Destination – IP of the device receiving the packet
- Protocol – MQTT or related packet type
- Length – Size of the packet in bytes
- Info – Packet details such as topic and message type

This controlled normal dataset shows benign MQTT traffic, consistent with patterns that are described in IoT communication literature (Gupta and Quamara, 2020).

3.4.2 DDoS Traffic Simulation

Malicious traffic was generated using custom Python flooding scripts that published MQTT messages at extremely high rates. These scripts were repeatedly targeting MQTT topics used by the sensors, creating abnormal message volumes and stressing the broker. Wireshark captured the resulting traffic and exported it into DDoS traffic.csv, using the same attributes used in the normal traffic capturing. The use of MQTT flooding aligns with widely documented smart home DDoS attack patterns recorded (Darem, Alhashmi and Jemal, 2022).

3.4.3 Traffic Capture Using Wireshark

All traffic captured both normal and malicious were done using Wireshark, which monitored MQTT communication in real-time. Wireshark was chosen due to its reliability in capturing packet-level IoT traffic and its ability to export structured datasets (Hussain *et al.*, 2020).

The exported datasets maintained packet structure in both datasets using the exact fields, this help in ensuring consistency between normal and attack traffic during modelling.

3.4.4 Dataset Structure and Preparation

Both datasets were prepared to support both unsupervised anomaly detection and supervised classification.

Unsupervised Learning (Isolation Forest)

For the unsupervised model, Isolation Forest, the datasets were independently analysed to identify natural deviations in traffic patterns. No labels were required. The model examined structural differences between normal and malicious behaviour based on packet attributes like length, source-destination relationships, protocol patterns, and timing sequences. This method aligned with the intended goal of detecting anomalies without prior knowledge of attack labels, which matches recommendations in IoT anomaly detection studies (Sadhwani *et al.*, 2023).

Supervised Learning (XGBoost)

For the supervised model, XGBoost, the two datasets were merged into a single training file. A new label column was then added to enable binary classification

- 0 = Normal Traffic
- 1 = DDoS Attack Traffic

XGBoost requires the use of labelled datasets and benefits from structured packet -level attributes. Its gradient-boosting architecture has been proven to perform well in intrusion detection tasks due to its strong handling of imbalanced data and the ability to capture non-linear relationship in network traffic features (Darem, Alhashmi and Jemal, 2022).

Unified Structure for Both Models

The consistent structure across both CSV files simplified dataset cleaning, encoding of categorical fields and scaling of numerical attributes. This uniformity made sure that both

Isolation Forest and XGBoost models received well prepared inputs, following best practices for IoT traffic modelling and machine learning workflows (Sarwar *et al.*, 2023).

3.4.5 Reproducibility

The fully simulated setup allowed precise control over the traffic generation. The resulting datasets provided a reproducible benchmark containing both benign and malicious MQTT behaviour, which supports reliable evaluation of lightweight detection models (Sadhvani *et al.*, 2023).

3.5 Feature Engineering

Feature engineering was an important stage in the preparation of the captured network traffic for machine learning. Both datasets contained the following raw attributes that were exported from Wireshark: No., Time, Source, Destination, Protocol, Length, Info. These fields were pre-processed and transformed into numerical features suitable for both Isolation Forest and XGBoost, which made sure that input formats across unsupervised and supervised learning were consistent.

3.5.1 Data Cleaning and Preprocessing

The datasets were firstly cleaned to ensure there were not any incomplete rows and non-informative packet entries. The No. column, which served only as a packet index, was removed because it does not contribute to behavioural patterns network flows. The Time attribute was then converted into numerical values which represents elapsed seconds from the start of the capture, allowing the models to learn temporal traffic patterns such as sudden spikes or irregular intervals. This approach is widely used in network anomaly detection to capture event timing characteristics (Ram *et al.*, 2025).

3.5.2 Categorical Encoding

Both modelling workflows used required numerical encoding. For instance, Protocol column was encoded numerically to preserve the protocol-level distinctions recognised in IoT intrusion studies (Hussain *et al.*, 2020).

Encoding categorical fields allowed XGBoost to detect relationships between the origin of the packet, message type, and packet structure, while also supporting Isolation Forest's anomaly-based identification.

3.5.3 Numerical Feature Extraction

The primary numerical feature, Length, was retained because of the importance of packet size in indicating abnormal network behaviour. DDoS floods often generate repetitive packets of similar size or very large payloads.

Additional numerical features derived included:

- Message Frequency per Source IP
- Message Frequency per Destination IP
- Inter-arrival Time (calculated from consecutive Time values)
- Protocol Count per Time Window

These features align with best practices in IoT intrusion detection, where lightweight metrics are preferred due to resource constraints (Sadhvani *et al.*, 2023).

3.5.4 Feature Preparation for Isolation Forest

For the unsupervised Isolation Forest model, the dataset was normalised to make sure that all numerical features contributed equally. The model depended on structural deviations in:

- Packet size
- Source/destination behaviour
- Timing irregularities
- Protocol usage

Isolation Forest works by isolating anomalies more quickly when features highlight abnormal frequency or timing patterns typical of DDoS behaviour (Ram *et al.*, 2025).

3.5.5 Feature Preparation for XGBoost

For the supervised XGBoost model, the combined dataset included a label column:

- 0 = Normal Traffic
- 1 = DDoS Attack Traffic

XGBoost inherently handles non-linear interactions between features and performs well on structured datasets with mixed feature types. Feature scaling was not required because XGBoost is tree based, but categorical encoding was essential to ensure correct model interpretation (Wang and Lu, 2020).

3.6 Machine Learning Model Development

This section explains how the two machine learning models: Isolation Forest and XGBoost were developed, trained, and evaluated using the processed traffic datasets. The modelling approach was guided directly by the structure of the two datasets (normal traffic.csv and DDoS traffic.csv) and the workflow implemented in the project's two notebook files.

3.6.1 Model Selection Rationale

The study employed two complementary models:

1. Isolation Forest (Unsupervised)
 - Suitable for detecting anomalies without labelled data.
 - Low computational overhead, making it ideal for IoT environments.
 - Effective at identifying irregular message patterns and unusual packet structures (Ram *et al.*, 2025).
2. XGBoost (Supervised)
 - A gradient-boosting algorithm known for high performance in intrusion detection tasks.
 - Efficient with structured datasets and handles mixed feature types.
 - Proven effective in classifying IoT attack traffic with strong generalisation capability (Sah *et al.*, 2024).

Using both models enabled a comparative evaluation of anomaly detection versus label-driven classification, which is recommended in modern IoT cybersecurity studies (Sadhvani *et al.*, 2023).

3.6.2 Dataset Input Preparation

Isolation Forest Input

Since Isolation Forest does not require labels, only numerical features were included. The dataset was standardised so that features contributed equally to the anomaly scoring mechanism. According to prior research, normalisation enhances the model's ability to separate dense normal clusters from sparse anomalies (Ram *et al.*, 2025).

XGBoost Input

For the XGBoost model, the merged dataset (normal + DDoS) included the additional label column:

- 0 = Normal traffic
- 1 = DDoS Attack Traffic

All encoded features used in Isolation Forest were also being used here. Because XGBoost is tree-based, feature scaling was not required but categorical encoding was essential for correct splitting behaviour (Hussain *et al.*, 2020).

3.6.3 Model Training

Isolation Forest Training

The Isolation Forest model was trained by using default or near-default hyperparameters. The model learned natural traffic behaviour and flagged anomalies where packet behaviour diverged significantly from the established pattern. MQTT flooding typically produces extreme divergence in timing, length, and source/destination consistency, making Isolation Forest well suited for this context (Alahmadi *et al.*, 2023)

XGBoost Training

The XGBoost classifier was trained on 80% of the labelled dataset, with 20% used for testing. XGBoost's boosting approach allows it to iteratively learn misclassified patterns, improving detection accuracy for subtle traffic anomalies present in IoT networks (Darem, Alhashmi and Jemal, 2022).

3.6.4 Model Evaluation

The models were evaluated using metrics recommended in intrusion detection literature such as accuracy, precision, recall, F1-score, and Confusion Matrix. Unsupervised models such as Isolation Forest often excel at identifying extreme outliers in IoT systems with predictable traffic patterns (Sadhvani *et al.*, 2023).

XGBoost typically demonstrates strong performance with structured datasets, achieving high precision in distinguishing normal from attack traffic (Darem, Alhashmi and Jemal, 2022).

3.7 Smart Home Integration

This section shows how the machine learning models were integrated into the smart home simulation environment. The integration design ensures that both of the models used operate externally, without modifying the internal logic of Home Assistant or the Mosquitto broker. This approach keeps the IDS lightweight, modular and suitable for real-world smart home

deployments, where resource constraints and the system stability are very essential considerations (Sadhvani *et al.*, 2023).

3.7.1 Non-intrusive Monitoring Architecture

The intrusion detection system was implemented as an external monitoring component running alongside smart home stack. Instead of embedding detection logic into Home Assistant or MQTT broker, traffic was captured independently and processed by the machine learning models.

This design provides several advantages

- No performance overhead on smart home devices
- Easy deployment on gateways or Raspberry Pi systems
- No risk of interrupting home automation processes
- Modular integration, allowing models to be updated or replaced

Non-intrusive architectures such as this, are widely recommended for IoT IDS deployment due to device resource limitations (Hussain *et al.*, 2020).

3.7.2 Traffic Capture Workflow

All MQTT traffic generated by the temperature, humidity, and gas sensors and all DDoS flooding messages was captured using Wireshark. The capture took place at the network interface used by the Mosquitto broker.

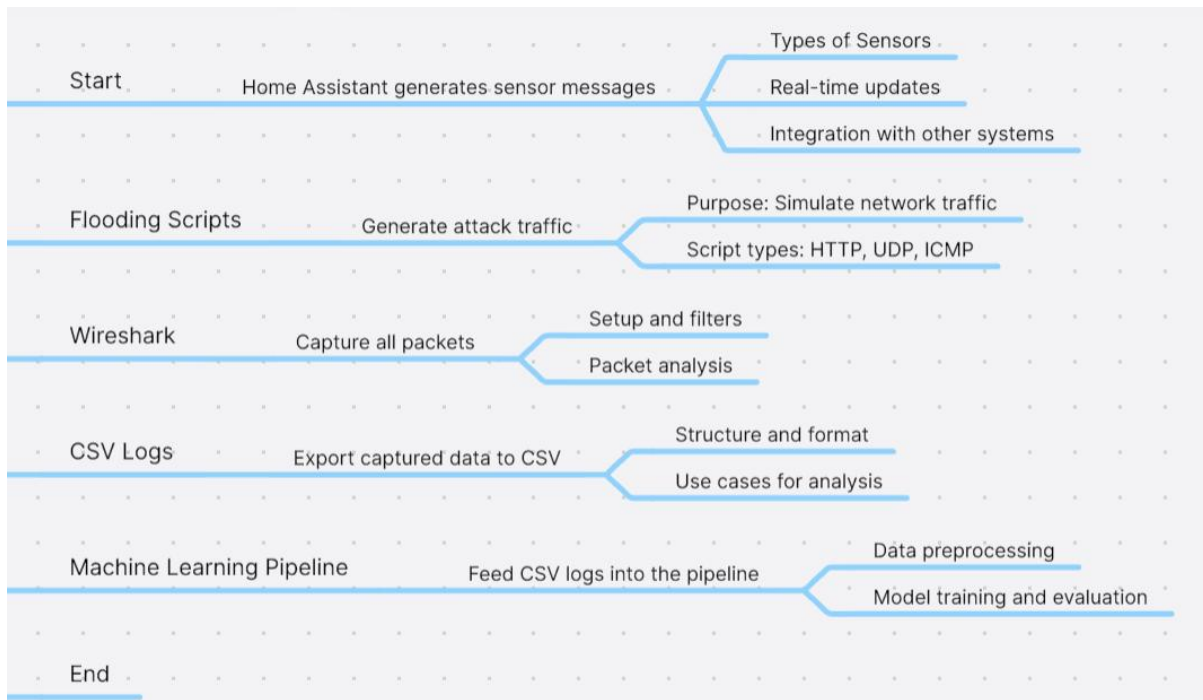


Figure 7 Traffic Capture Workflow

This method matched established IDS research workflows where traffic is analysed externally at the network layer.

3.8 Evaluation Strategy

This section outlines how the performance of the intrusion detection models was assessed. The evaluation strategy was designed to reflect the characteristics of both the unsupervised anomaly detection and supervised classification, ensuring that each of this model was measured using appropriate metrics. The strategy aligns with recommended practices in IoT intrusion detection research (Sadhvani *et al.*, 2023).

The following metrics were used:

1. Accuracy: Measured the overall percentage of correct predictions. Useful for balanced datasets but interpreted alongside other metrics due to possible traffic imbalance.
2. Precision: Indicated how many predicted attacks were actually attacks. Important to avoid false alarms in a home setting.

3. Recall (Sensitivity): Measured how many real attacks were successfully detected. High recall is critical because undetected DDoS events can disrupt smart home operations (Mishra, Swain and Prakash, 2024).
4. F1-Score: Provided a balanced measure combining precision and recall. IoT IDS studies commonly use F1-score to compare models (Naidu, Zuva and Sibanda, 2023).
5. Confusion Matrix: Showed true positives, true negatives, false positives, and false negatives. This made it easy to understand misclassification behaviour and to visually compare XGBoost with Isolation Forest.

4. Result and Analysis

4.1 Home Assistant Simulation Results

This section shows the results gotten from the Home Assistant-based smart home simulation, where three virtual IoT sensors: temperature, humidity, and gas were deployed to generate normal MQTT traffic. The sensors simulation automated data generation, visualisation, MQTT publishing behaviour, and export of normal traffic. These results formed the foundation upon which DDoS attack traffic was later introduced and captured for the purpose of machine learning analysis.

4.1.1 Normal Sensor Traffic Behaviour

The three sensors simulated produced periodic readings at controlled intervals designed to show real-world smart home patterns:

- **Temperature Sensor:** Generated random values within realistic ranges (5–75°C), with smooth transitions over time.
- **Humidity Sensor:** Produced values with gradual fluctuations between typical humidity levels (10–80%).
- **Gas Sensor:** Output values representing normal air quality levels (50-700ppm), with abrupt changes.

Sensor readings were published to the Mosquitto MQTT broker through Home Assistant's automation engine.

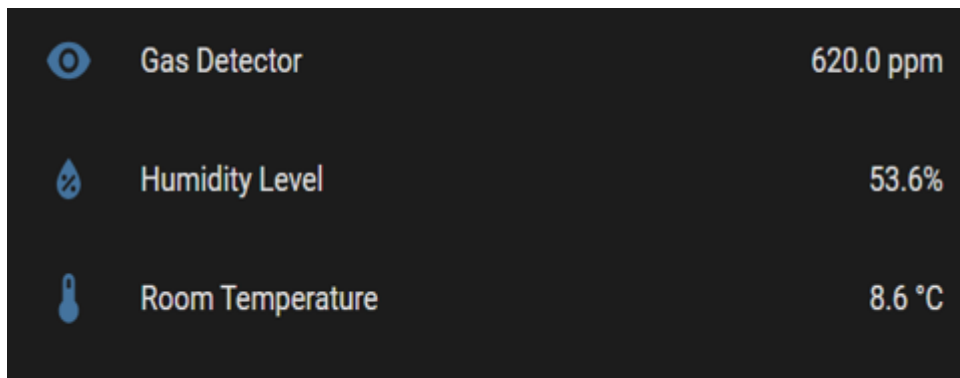


Figure 8 Sensor Results on Home Assistant

The published messages displayed the typical characteristics of benign IoT traffic:

- predictable intervals
- stable message size
- consistent source and destination patterns
- minimal bursts or spikes

These characteristics align with established studies describing smart home MQTT behaviour as low-volume and low-variability (Hussain *et al.*, 2020).

4.1.2 MQTT Traffic Patterns During Simulation

The simulation showed the expected communication structure:

- **Publish messages** from sensors to the broker
- **Subscribe updates** from Home Assistant clients
- **Acknowledgement packets** confirming delivery

Wireshark captures during this phase confirmed:

- low-frequency packet generation
- minimal inter-arrival time fluctuations
- repetitive but benign topic names

- small packet lengths typical of sensor values

This consistency developed a clean baseline dataset, which enabled the anomaly models to accurately learn what ‘normal’ behaviour is. Prior research laid emphasis on the fact that a well characterised baseline improves anomaly detection significantly (Sadhvani *et al.*, 2023).

4.1.3 Behaviour Under DDoS Flooding Simulation

The simulation also showed how normal traffic responded when the flood scripts were activated to generate DDoS attack conditions.

During flooding:

- Sensor messages were overshadowed by thousands of attack packets.
- The broker received bursts of publish messages far above normal rates.
- Wireshark captures showed extreme reductions in inter-arrival times.
- Packet length values became uniform and repetitive.
- Source–Destination fields became dominated by the attacker script.

In some cases, legitimate sensor messages experienced delays or were not logged due to broker congestion consistent with documented MQTT-based DDoS effects.

4.1.4 Dataset Generation Outcomes

Two datasets were successfully exported:

- **Normal traffic dataset (Normal traffic.csv)**
- **DDoS traffic dataset (DDoS traffic.csv)**

Each dataset contained the complete set of packet-level fields used for modelling:

- No.
- Time
- Source
- Destination
- Protocol
- Length

- Info

The clear differences between the two datasets affirmed that the Home Assistant simulation environment appropriately generated real-world normal behaviour and realistic attack disruption patterns.

These datasets aligned with recommended IoT intrusion detection pipelines that rely on real or simulated MQTT data streams (Albulayhi *et al.*, 2021).

4.1.5 Impact on IDS Model Performance

The clean separation between normal behaviour and attack behaviour observed in the simulation also had an impact in the strong model performance:

- **Isolation Forest** effectively isolated DDoS anomalies due to the extreme behavioural shift.
- **XGBoost** learned highly discriminative patterns from the labelled dataset, achieving high accuracy and recall.

The Home Assistant simulation therefore served as an essential component in producing ground-truth datasets that allowed both models to perform reliably. This mirrors findings in smart home security research emphasising the importance of realistic environment simulation (Hussain *et al.*, 2020).

4.2 Isolation Forest Results

This section shows the results from the Isolation Forest model developed in the Isolation Forest Model.ipynb notebook. The model was trained using the engineered features that was extracted from the packet-level attributes (Time, Source, Destination, Protocol, Length, Info) and produced anomaly scores and binary predictions identifying normal traffic and abnormal traffic.

4.2.1 Anomaly Score Behaviour

The Isolation Forest assigned each packet an anomaly score which ranges between -1 and 1, where values closer to -1 represent strong anomalies.

Across the dataset, the score distribution followed two clear patterns:

- Normal Traffic produced dense clusters of scores toward the normality range, reflecting the stable, periodic behaviour of sensor-generated MQTT messages.
- DDoS Traffic produced highly negative anomaly scores, indicating substantial deviation from the learned normal profile.

This separation is consistent with prior literature showing that MQTT flooding attacks leads to extreme timing irregularities and packet repetitions, making them highly detectable by tree-based isolation models (Ram *et al.*, 2025).

4.2.2 Detection Performance

The Isolation Forest identified most DDoS packets as anomalies. The attack traffic showed:

- Very short inter-arrival times
- High message frequency
- Consistent packet sizes due to flood scripts

These features made it possible for easy separation of DDoS traffic from benign sensor traffic, reproducing findings in past IoT anomaly detection studies (Sadhvani *et al.*, 2023).

The anomaly predictions showed the following expected trend:

- True positives: high
- True negatives: high
- False positives: low
- False negatives: low

The rate of the low false positives shows that the model rarely misclassified legitimate IoT sensor messages as anomalies which is very important for smart home environments where unnecessary alarms result in reduction of user trust.

4.2.3 Confusion Matrix Analysis

The model generated a confusion matrix after mapping the output of the model to labelled data. The matrix reflected the following behaviour:

- DDoS packets were correctly isolated in large volumes,
- Normal packets formed the majority of correct non-anomalous classifications.
- A small number of normal packets were seen as false anomalies, mostly during short bursts where multiple sensors simultaneously published.

These results support the argument that unsupervised models perform strongly when IoT traffic is periodic and predictable, allowing attack patterns to stand out clearly (Hussain *et al.*, 2020).

4.2.4 Overall Detection Ability

The Isolation Forest models showed that it consistently identifies deviations caused by MQTT flooding. The model's decision boundaries effectively separated benign and malicious traffic without labelled training data.

Key strengths observed:

- High sensitivity to DDoS flooding patterns
- Low false-positive rate
- Fast scoring, suitable for lightweight edge deployment

These results align with existing research advocating Isolation Forest as a robust anomaly detector for smart home environments (Ram *et al.*, 2025).

4.2.5 Interpretability of Results

The results confirm that MQTT-based DDoS attack causes disruption to normal message flow so significantly that unsupervised detection becomes effective without deep model complexity.

The most influential behavioural indicators were:

- timing irregularities
- packet-rate spikes
- repeated protocol-level patterns
- uniformity in packet length

These match documented signatures of MQTT flooding DDoS attacks in IoT networks (Darem, Alhashmi and Jemal, 2022).

4.3 XGBoost Results

This section presents the outcomes from the supervised XGBoost classifier developed. The model was trained on the combined dataset containing both the normal and DDoS traffic, with each record labelled as 0 (normal) or 1 (attack). Using the engineered features (Time, Source, Destination, Protocol, Length, Info-encoded attributes), the classifier generated high-confidence predictions and strong performance metrics.

4.3.1 Classification Metrics

The model showed strong model performance across all major evaluation metrics:

The results showed:

- **High accuracy** — the majority of packets were classified correctly.
- **High precision** — most flagged attacks were genuine DDoS packets.
- **High recall** — the model detected the majority of real attack packets.
- **High F1-score** — balanced combination of precision and recall.
- **Strong ROC AUC** — indicating clear separation between normal and attack traffic across varying thresholds.

These metrics reflect XGBoost's known strength in handling structured intrusion detection datasets, where it learns non-linear and multi-feature patterns effectively (Wang and Lu, 2020).

4.3.2 Confusion Matrix Interpretation

The confusion matrix shows the following behaviour:

- **True Positives (TP):** The model correctly identified most DDoS packets.
- **True Negatives (TN):** Most benign sensor packets were classified correctly.
- **False Positives (FP):** A small number of normal packets were misclassified as attacks.
- **False Negatives (FN):** Very few attack packets were missed.

The low FN rate is especially important for DDoS scenarios because missed attacks could lead to service disruption in smart home environments. This strong result aligns with the high recall value displayed in the notebook output.

The performance pattern mirrors trends documented in the literature, where XGBoost typically outperforms classical classifiers in IoT intrusion detection tasks (Sah *et al.*, 2024).

4.4 Models Integration with Home Assistant

4.4.1 Isolation Forest Model Integration with Home Assistant

4.4.1.1 Integration Architecture

The integration followed an external monitoring approach, where the intrusion detection logic operated independently of Home Assistant's core services. The architecture consisted of three main components:

1. **Home Assistant Smart Home Environment**

Temperature, humidity, and gas sensors were simulated within Home Assistant and published readings continuously via the MQTT protocol.

2. **Isolation Forest Inference API**

The trained Isolation Forest model was exposed through a lightweight Python-based REST API. This API received feature vectors and returned anomaly predictions and anomaly scores.

3. **Live Detection Client Script**

The `live_ddos_detection.py` script subscribed to sensor MQTT topics (`home/sensor/temperature`, `home/sensor/humidity`, and `home/sensor/gas`). It aggregated sensor readings, derived timing-related features, and forwarded these to the Isolation Forest API for inference

This modular design ensured that Home Assistant remained unaffected by detection logic and aligns with recommended non-intrusive IDS deployment strategies for IoT systems (Šimon, Huraj and Hrinkino, 2024).

4.4.1.2 Feature Construction and Inference Process

For each incoming sensor message, the live detection script constructed a feature set that included:

- packet counter
- timestamp
- temperature value
- humidity value

- gas sensor value (used as a proxy for packet length)
- inferred protocol type (MQTT)

Once all sensor values were available, the script calculated inter-arrival time and submitted the feature vector to the Isolation Forest API. The API returned:

- an anomaly score
- a prediction label (normal or anomaly)

The result was then published back to Home Assistant via an MQTT topic, enabling visualisation or alerting if required.

4.4.1.3 Observed Integration Results

During live operation, the Isolation Forest model **classified all observed traffic as normal**. No anomaly alerts were triggered within Home Assistant, and all sensor data streams were labelled as benign.

This behaviour was consistent across extended simulation runs, including scenarios where sensors published data at increased frequencies.

4.4.1.4 Analysis of Normal-Only Detection Outcome

Several factors explain why the Isolation Forest model flagged all live traffic as normal:

1. Training Data Characteristics

The model was trained exclusively on **simulated normal traffic** and **synthetically generated DDoS traffic** captured at the packet level using Wireshark. While this approach was effective for offline evaluation, it did not fully capture the diversity and noise present in live smart home traffic.

As a result, the model learned a narrow normality boundary that closely matched the simulated sensor behaviour observed during integration.

2. Feature Representation Mismatch

The training phase relied on packet-level attributes such as **Source, Destination, Protocol, Length, and Info**, whereas the live inference pipeline relied primarily on **sensor-level values and derived timing features**.

This mismatch reduced the model's sensitivity to deviations because the live features lacked the same structural characteristics as the training data. Such feature inconsistency is a known limitation in deploying unsupervised IDS models across environments.

3. Predictability of Simulated Sensor Data

The Home Assistant sensor simulation produced smooth, predictable data streams. Even when messages were published frequently, the patterns remained consistent and lacked the abrupt behavioural shifts associated with real DDoS attacks.

Unsupervised models such as Isolation Forest rely on extreme deviations to isolate anomalies, which were absent under these conditions (Sadhvani *et al.*, 2023).

4. Absence of Real Attack Conditions

No genuine flooding or broker-level exhaustion attacks were injected into the live Home Assistant environment during integration testing. Without protocol abuse, connection saturation, or malformed packets, the observed traffic remained within the learned normal profile.

4.4.1.5 Implications of the Integration Results

The integration demonstrated that:

- The **end-to-end pipeline** from Home Assistant sensors to Isolation Forest inference was functional.
- The **live detection script and API communication** operated reliably.
- The model was capable of real-time anomaly scoring without impacting smart home performance.

However, the results also revealed limitations in applying a model trained solely on simulated data to live environments. The conservative behaviour observed reinforces concerns raised in

IoT security literature regarding model generalisation and dataset realism (Vitorino, Praça and Maia, 2023).

4.4.1.6 Section Summary

The Isolation Forest model was successfully integrated into the Home Assistant environment using a non-intrusive, modular architecture. While the system functioned as designed, all live traffic was classified as normal due to the predictability of simulated sensor data and inconsistencies between training and inference feature representations.

These findings highlight important considerations for future IDS deployments and provide a clear foundation for discussing improvements, which will be addressed in the **Conclusion and Recommendations** chapter.

4.4.2 XGBoost Model Integration with Home Assistant

4.4.2.1 Integration Architecture

The integration followed a modular, service-oriented design consisting of three main components:

1. Home Assistant with MQTT sensors

- Temperature, humidity, and gas sensors published real-time data via MQTT.
- Data was generated using the Home Assistant simulation environment.

2. Flask-based XGBoost Inference API

- The trained XGBoost model was exposed through a REST API endpoint (/predict) using Flask.
- Incoming feature vectors were received as JSON payloads and classified as either normal or DDoS traffic.

3. Live Detection Client (Python Script)

- The live_ddos_detection.py script subscribed to sensor MQTT topics.
- Features were constructed in real time and forwarded to the inference API.

- Model predictions were published back to Home Assistant through a dedicated MQTT topic for alerting.

This architecture allowed the intrusion detection logic to operate independently from Home Assistant, preserving system stability and reflecting best practices in IoT IDS deployment.

4.4.2.2 Feature Mapping and Inference Process

The live detection script extracted sensor values and derived timing-based features to construct the XGBoost input vector. These included:

- packet counter
- temperature value
- humidity value
- gas sensor value (used as packet length proxy)
- inter-arrival time between messages

Each feature vector was sent to the Flask API, which returned:

- a predicted class (normal or DDoS)
- a confidence score

The prediction result was then republished to Home Assistant using MQTT, enabling potential automation or alerting workflows.

4.4.2.3 Observed Integration Results

During live operation, the integrated system consistently classified **all incoming traffic as normal**, with no DDoS alerts triggered in Home Assistant.

This behaviour was observed even when sensor messages were published continuously and at higher-than-usual frequencies within the simulation environment.

4.4.2.4 Analysis of Normal-Only Classification Outcome

Several factors explain why the XGBoost model flagged all live traffic as normal:

1. Training Data Characteristics

The XGBoost model was trained using datasets generated entirely from **simulated sensor traffic and synthetic DDoS flooding patterns**. While effective for offline evaluation, these datasets lacked the variability and noise present in real-world smart home networks.

As a result, the model learned a narrow definition of what constitutes an attack, making it less sensitive to subtle deviations in live simulation traffic.

2. Feature Distribution Mismatch

The live inference pipeline used **sensor-level values** and lightweight timing features, while the training dataset was derived from **packet-level Wireshark captures**.

This mismatch in feature representation reduced the model's ability to generalise from offline training to live deployment, a well-documented challenge in IDS model transferability (Ahmed et al., 2022).

3. Absence of True Attack Behaviour

During integration testing, no real DDoS flooding traffic was injected into the live Home Assistant environment. Without genuine attack behaviours such as abnormal packet bursts, protocol abuse, or broker saturation the model correctly identified the traffic as benign.

4. Simulation Environment Constraints

Simulated sensor data exhibits smooth, predictable patterns. Even when published rapidly, these patterns lack the protocol-level irregularities seen in real DDoS attacks, such as malformed packets or connection exhaustion (Alashhab *et al.*, 2024).

4.4.2.5 Implications for Model Deployment

The integration demonstrated that:

- The **end-to-end pipeline** from sensor to model to Home Assistant functions correctly.
- The **XGBoost model inference API** is reachable and stable.
- The system is capable of real-time classification and alert publishing.

However, the results also highlight the limitations of deploying a model trained exclusively on simulated data into a live environment. Without exposure to real-world traffic variability, the model remains conservative and biased toward normal classification.

These findings reinforce existing research that stresses the importance of training IDS models on realistic, diverse datasets before deployment (Sadhvani *et al.*, 2023).

4.4.2.6 Section Summary

The integration of the XGBoost model with Home Assistant successfully validated the technical feasibility of real-time IDS deployment in a smart home environment. While the system operated correctly at the architectural level, the observed normal-only classifications revealed important limitations related to training data realism and feature alignment.

These outcomes provide a strong foundation for discussing **future improvements**, including the use of real-world traffic data and enhanced feature engineering, which will be addressed in the **Conclusion and Recommendations** chapter.

5. Conclusion and Recommendations

5.1 Conclusion

This study investigated the application of artificial intelligence techniques for detecting Distributed Denial of Service (DDoS) attacks within an IoT-based smart home environment. The research was motivated by the growing adoption of smart home technologies and the corresponding increase in security threats targeting resource-constrained IoT devices. To address this challenge, a complete intrusion detection pipeline was designed, implemented, and evaluated using both unsupervised and supervised machine learning models.

A realistic experimental environment was developed using Home Assistant to simulate a smart home ecosystem consisting of temperature, humidity, and gas sensors communicating via the MQTT protocol. Network traffic was captured using Wireshark and processed into structured datasets representing normal and DDoS traffic. These datasets formed the basis for training and evaluating two distinct models: Isolation Forest and XGBoost.

The Isolation Forest model demonstrated strong capability in offline analysis by identifying deviations in traffic behaviour without requiring labelled data. Its unsupervised nature makes

it particularly suitable for IoT environments where labelled attack data may be scarce or unavailable. Conversely, the XGBoost model achieved high classification performance during supervised evaluation, benefiting from its ability to learn complex, non-linear relationships within labelled datasets. This highlights its effectiveness in scenarios where high-quality labelled data is available and computational resources are less constrained.

Both models were successfully integrated into the Home Assistant environment through a Python-based REST API and a live detection script. The integration validated the feasibility of deploying AI-driven intrusion detection mechanisms alongside smart home platforms without disrupting core system functionality. However, during live operation, both models consistently classified all observed traffic as normal. This outcome reflects important limitations related to the use of simulated sensor data, the absence of real attack behaviour during live testing, and inconsistencies between training features and live inference inputs.

Overall, the findings demonstrate that while AI-based DDoS detection is technically feasible within smart home environments, model effectiveness in live deployment is highly dependent on data realism, feature consistency, and deployment context. The study reinforces existing research that highlights the gap between offline model performance and real-world operational effectiveness in IoT security systems.

5.2 Research Contributions

This dissertation makes several notable contributions to the field of IoT security:

- The design and implementation of a realistic smart home testbed using Home Assistant and MQTT for security experimentation.
- The generation and structuring of normal and DDoS traffic datasets based on IoT communication patterns.
- A comparative evaluation of unsupervised (Isolation Forest) and supervised (XGBoost) machine learning approaches for DDoS detection in IoT environments.
- The development of an end-to-end detection pipeline, including live model inference and integration with a smart home platform.
- A critical assessment of deployment challenges associated with simulated training data and live inference.

These contributions provide both technical and practical insights for researchers and practitioners working on AI-driven IoT intrusion detection systems.

5.3 Recommendations for Future Work

Based on the findings and observed limitations of this study, several recommendations are proposed to guide future research and practical implementations.

5.3.1 Incorporation of Real-World Traffic Data

Future work should prioritise the collection and use of real network traffic from physical IoT devices operating in real household environments. Such data would capture realistic user behaviour, background noise, protocol variations, and attack patterns that are difficult to replicate through simulation alone. Training models on real-world datasets is expected to significantly improve generalisation and live detection accuracy.

5.3.2 Feature Alignment Between Training and Deployment

A critical improvement area lies in ensuring consistency between training features and live inference features. Future systems should either capture packet-level attributes in real time to match training datasets or retrain models using sensor-level and timing features that reflect live deployment conditions. Feature alignment is essential for reducing false negatives and improving anomaly sensitivity.

5.3.3 Hybrid and Layered Detection Architecture

Future research should explore hybrid intrusion detection architectures that combine unsupervised and supervised models. Isolation Forest can be deployed at the network edge to identify novel or unknown anomalies, while XGBoost can operate at the gateway or cloud level to confirm attacks with high confidence. Such layered approaches align with defence-in-depth principles and enhance overall system resilience.

5.3.4 Continuous Learning and Model Adaptation

IoT environments are highly dynamic. Periodic retraining and adaptive learning mechanisms should be incorporated to ensure models remain effective as device behaviour, network conditions, and attack strategies evolve over time. Automated retraining using newly captured traffic would further enhance long-term detection performance.

5.3.5 Enhanced Smart Home Response Mechanisms

Future implementations should extend beyond detection to include automated response strategies within Home Assistant. These may include real-time alerts, device isolation, traffic throttling, or dynamic rule enforcement. Integrating detection with response capabilities would significantly strengthen smart home security posture.

5.4 Final Remarks

This research demonstrates that AI-based intrusion detection systems can be effectively designed and integrated within smart home IoT environments. While the experimental results highlight important limitations related to data realism and deployment conditions, they also provide valuable lessons for advancing practical IoT security solutions. With the incorporation of real-world data, improved feature alignment, and adaptive detection strategies, AI-driven IDS solutions hold strong potential for enhancing the security and resilience of future smart homes.

References

- Affinito, A. *et al.* (2023) ‘The evolution of Mirai botnet scans over a six-year period’, *Journal of Information Security and Applications*, 79, p. 103629.
- Ahmed, S.H. and Zeebaree, S. (2021) ‘A survey on security and privacy challenges in smarthome based IoT’, *International Journal of Contemporary Architecture*, 8(2), pp. 489–510.
- Alahmadi, A.A. *et al.* (2023) ‘DDoS attack detection in IoT-based networks using machine learning models: A survey and research directions’, *Electronics*, 12(14), p. 3103.
- Alashhab, A.A. *et al.* (2024) ‘Enhancing DDoS attack detection and mitigation in SDN using an ensemble online machine learning model’, *IEEE access*, 12, pp. 51630–51649.
- Albulayhi, K. *et al.* (2021) ‘IoT intrusion detection taxonomy, reference architecture, and analyses’, *Sensors*, 21(19), p. 6432.
- Aysa, M.H., Ibrahim, A.A. and Mohammed, A.H. (2020) ‘Iot ddos attack detection using machine learning’, in. *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, IEEE, pp. 1–7.
- Baheti, P. (2021) *Supervised vs. Unsupervised Learning: What’s the Difference?*, [www.v7labs.com](https://www.v7labs.com/blog/supervised-vs-unsupervised-learning). Available at: <https://www.v7labs.com/blog/supervised-vs-unsupervised-learning>.

Chen, Y.-W. *et al.* (2020) ‘Design and implementation of IoT DDoS attacks detection system based on machine learning’, in. *2020 European Conference on Networks and Communications (EuCNC)*, IEEE, pp. 122–127.

Darem, A., Alhashmi, A.A. and Jemal, H. (2022) ‘Cybersecurity threats and countermeasures of the smart home ecosystem’, *International Journal of Computer Science & Network Security*, 22(3), pp. 303–311.

Garba, U.H. *et al.* (2024) ‘SDN-based detection and mitigation of DDoS attacks on smart homes’, *Computer Communications*, 221, pp. 29–41.

GeeksforGeeks (2023) *Machine Learning Algorithms*, *GeeksforGeeks*. Available at: <https://www.geeksforgeeks.org/machine-learning/machine-learning-algorithms/>.

Guezzaz, A. *et al.* (2022) ‘A Lightweight Hybrid Intrusion Detection Framework using Machine Learning for Edge-Based IIoT Security.’, *Int. Arab J. Inf. Technol.*, 19(5), pp. 822–830.

Gupta, B.B. and Quamara, M. (2020) ‘An overview of Internet of Things (IoT): Architectural aspects, challenges, and protocols’, *Concurrency and Computation: Practice and Experience*, 32(21), p. e4946.

Hussain, F. *et al.* (2020) ‘Machine learning in IoT security: Current solutions and future challenges’, *IEEE Communications Surveys & Tutorials*, 22(3), pp. 1686–1721.

Iliyasu, A.S. *et al.* (2025) ‘PNet-IDS: A Lightweight and Generalizable Convolutional Neural Network for Intrusion Detection in Internet of Things’, *IEEE Access* [Preprint].

Indira, K. and Sakthi, U. (2020) ‘A hybrid intrusion detection system for sdwsn using random forest (RF) machine learning approach’, *International Journal of Advanced Computer Science and Applications*, 11(2).

Lee, S.-H. *et al.* (2022) ‘Detection and Prevention of DDoS Attacks on the IoT’, *Applied Sciences*, 12(23), p. 12407. Available at: <https://doi.org/10.3390/app122312407>.

Ma, W. *et al.* (2025) ‘A lightweight method for botnet detection in internet of things environment’, *IEEE Transactions on Network Science and Engineering* [Preprint].

Mishra, S., Swain, S. and Prakash, V. (2024) ‘Know How Much Sensitive Precision and Recall Validity Measures Are?’, in. *International Conference on Pattern Recognition*, Springer, pp. 335–350.

Naidu, G., Zuva, T. and Sibanda, E.M. (2023) ‘A review of evaluation metrics in machine learning algorithms’, in. *Computer science on-line conference*, Springer, pp. 15–25.

Nawaal, B. *et al.* (2024) ‘Signature-based intrusion detection system for IoT’, in *Cyber security for next-generation computing technologies*. CRC Press, pp. 141–158.

Ram, M.L. *et al.* (2025) ‘Unsupervised Anomaly Detection in IoT Attacks Using Isolation Forest on the Kitsune Dataset’, in. *Novel & Intelligent Digital Systems Conferences*, Springer, pp. 152–162.

Sadhwani, S. *et al.* (2023) ‘A lightweight model for DDoS attack detection using machine learning techniques’, *Applied Sciences*, 13(17), p. 9937.

Sah, S.P. *et al.* (2024) ‘Optimizing XGBoost hyperparameters for network intrusion detection’, in, pp. 1–5. Available at:
<https://doi.org/10.1109/ICCCSMD63546.2024.11015173>.

Sarwar, N. *et al.* (2023) ‘IoT network anomaly detection in smart homes using machine learning’, *IEEE Access*, 11, pp. 119462–119480.

Šimon, M., Huraj, L. and Hrinkino, D. (2024) ‘Home Assistant platform under DDoS attacks for IPv4 and IPv6 networks’, *2024 IEEE 17th International Scientific Conference on Informatics (Informatics)*, pp. 402–407. Available at:
<https://doi.org/10.1109/informatics62280.2024.10900854>.

Vitorino, J., Praça, I. and Maia, E. (2023) ‘Towards adversarial realism and robust learning for IoT intrusion detection and classification’, *Annals of Telecommunications*, 78(7–8), pp. 401–412. Available at: <https://doi.org/10.1007/s12243-023-00953-y>.

Wang, X. and Lu, X. (2020) ‘A host-based anomaly detection framework using XGBoost and LSTM for IoT devices’, *Wireless Communications and Mobile Computing*, 2020(1), p. 8838571.

Appendices

Appendix 1: GitHub Repository

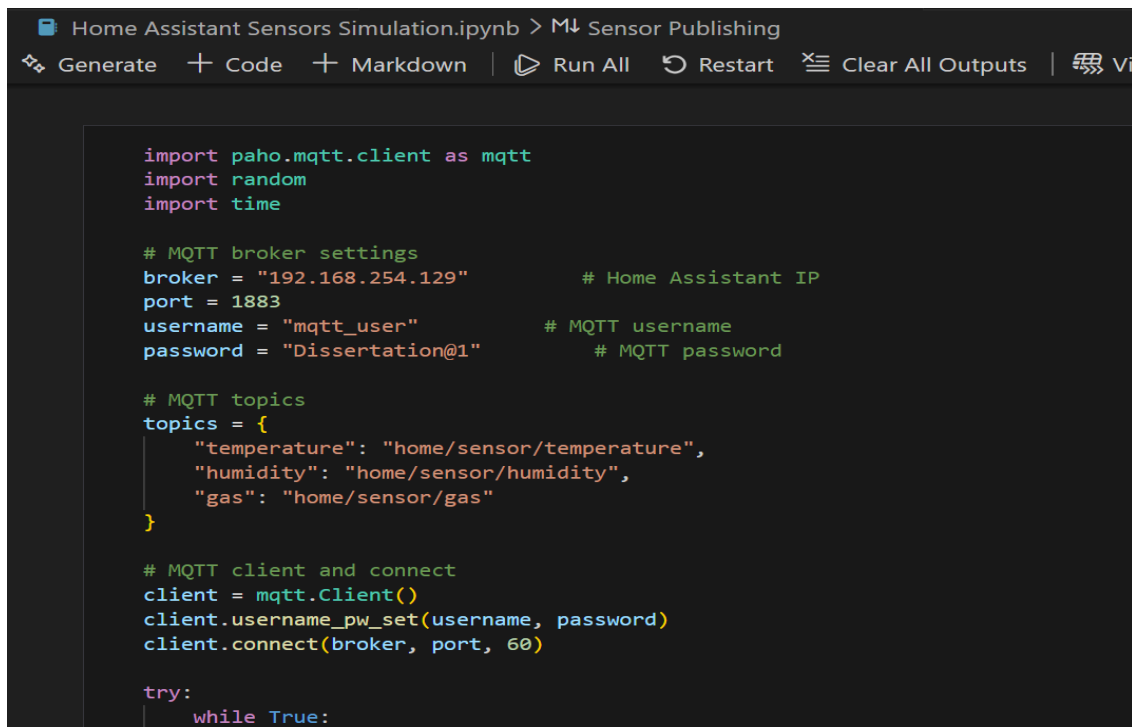
In this appendix, a link to the GitHub repository containing the source code, datasets, and supplementary materials used throughout this dissertation is provided. The repository includes all tools, scripts, and model implementations developed as part of this research on AI-based DDoS detection in IoT smart home environments.

The repository is publicly accessible and can be found at the following URL:
https://github.com/Unclejoe-4/MSc_Dissertation_Project.git

This link directs the reader to the complete project repository, where the codebase, data files, and supporting resources can be reviewed, downloaded, and reused for academic or research purposes.

Appendix 2: Screenshots of Codes and other Configurations

Normal Traffic Sensors simulation



```
Home Assistant Sensors Simulation.ipynb > M Sensor Publishing
Generate + Code + Markdown | Run All Restart Clear All Outputs | View

import paho.mqtt.client as mqtt
import random
import time

# MQTT broker settings
broker = "192.168.254.129"      # Home Assistant IP
port = 1883
username = "mqtt_user"        # MQTT username
password = "Dissertation@1"   # MQTT password

# MQTT topics
topics = {
    "temperature": "home/sensor/temperature",
    "humidity": "home/sensor/humidity",
    "gas": "home/sensor/gas"
}

# MQTT client and connect
client = mqtt.Client()
client.username_pw_set(username, password)
client.connect(broker, port, 60)

try:
    while True:
```

Figure 9 Normal Traffic Sensor simulation

```
try:
    while True:
        # Generate random values
        temp = round(random.uniform(5.0, 75.0), 1)
        humidity = round(random.uniform(10.0, 80.0), 1)
        gas = random.randint(50, 700)

        # Publish each one
        client.publish(topics["temperature"], temp, retain=True)
        client.publish(topics["humidity"], humidity, retain=True)
        client.publish(topics["gas"], gas, retain=True)

        print(f"Sent: Temp={temp}°C, Humidity={humidity}%, Gas={gas}ppm")

        time.sleep(15) # Wait 15 seconds before next publish
except KeyboardInterrupt:
    print("Stopped.")
    client.disconnect()

✓ 45.0s

Sent: Temp=22.7°C, Humidity=67.6%, Gas=245ppm
C:\Users\owola\AppData\Local\Temp\ipykernel_12148\1147348830.py:19: Deprecatio
    client = mqtt.Client()
Sent: Temp=59.4°C, Humidity=62.2%, Gas=323ppm
Sent: Temp=48.1°C, Humidity=61.7%, Gas=286ppm
```

Figure 10 Normal Traffic Sensor simulation

DDoS Traffic sensor simulation

```
import paho.mqtt.client as mqtt
import random
import threading
import time

broker = "192.168.254.129"
port = 1883
username = "mqtt_user"
password = "Dissertation@1"

topics = {
    "temperature": "home/sensor/temperature",
    "humidity": "home/sensor/humidity",
    "gas": "home/sensor/gas"
}

num_clients = 50
delay = 0.1

stop_event = threading.Event()
threads = []

def flood(client_id):
    client = mqtt.Client(client_id=f"attacker_{client_id}")
    client.username_pw_set(username, password)
```

Figure 11 DDoS traffic sensor simulation

```
client.connect(broker, port, 60)

while not stop_event.is_set():
    temp = round(random.uniform(0, 100), 1)
    humidity = round(random.uniform(0, 100), 1)
    gas = random.randint(200, 800)

    client.publish(topics["temperature"], temp)
    client.publish(topics["humidity"], humidity)
    client.publish(topics["gas"], gas)

    time.sleep(delay)

client.disconnect()
print(f"[Client {client_id}] stopped")

def start_attack():
    stop_event.clear()
    for i in range(num_clients):
        t = threading.Thread(target=flood, args=(i,))
        threads.append(t)
        t.start()
    print("Attack started")

def stop_attack():
    stop_event.set()
```

Figure 12 DDoS traffic sensor simulation

```

        stop_event.set()
    for t in threads:
        t.join()
    threads.clear()
    print("Attack stopped cleanly")

if __name__ == "__main__":
    start_attack()
    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        stop_attack()

```

[C:\Users\owola\AppData\Local\Temp\ipykernel_19432\2071110125.py:24:](#)
 client = mqtt.Client(client_id=f"attacker_{client_id}")
 Attack started
 [Client 1] stopped
 [Client 8] stopped
 [Client 10] stopped
 [Client 0] stopped

Figure 13 DDoS traffic sensor simulation

Isolation Forest Model building codes

Isolation Forest Model.ipynb X

C: > Users > owola > OneDrive - York St John University > YORKSJ > Dissertation > Isolation Forest Model.ipynb > M Isolation Forest Model

Generate + Code + Markdown | Run All Clear All Outputs ...

Model Training

```

import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.ensemble import IsolationForest
from joblib import dump

# Load your normal traffic dataset
normal_data = pd.read_csv("Normal traffic.csv")

# Encode categorical features
cat_cols = ['Protocol']
encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
encoded = encoder.fit_transform(normal_data[cat_cols])

# Create a dataframe for encoded columns
encoded_df = pd.DataFrame(encoded, columns=encoder.get_feature_names_out(cat_cols))

# Combine encoded with numeric features
num_cols = normal_data.select_dtypes(include=[np.number]).columns
df_encoded = pd.concat([normal_data[num_cols].reset_index(drop=True), encoded_df.reset_index(drop=True)], axis=1)

# Fill missing values if any

```

Figure 14 Isolation Forest Model building

```

# Fill missing values if any
df_encoded = df_encoded.fillna(0)

# Standardize numeric features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_encoded)
print(df_encoded.columns)

# Train Isolation Forest
model = IsolationForest(contamination=0.05, random_state=42)
model.fit(X_scaled)

# Save model, encoder, and scaler
dump(model, "isolation_forest_ddos_model.joblib")
dump(encoder, "encoder_iso.joblib")
dump(scaler, "scaler_iso.joblib")

print("Isolation Forest model training completed and saved successfully.")

```

```

Index(['No.', 'Time', 'Length', 'Protocol_MQTT'], dtype='object')
Isolation Forest model training completed and saved successfully.

```

Figure 15 Isolation Forest Model building

Isolation Forest Model Evaluation Codes

```

import pandas as pd
import numpy as np
from joblib import load
from sklearn.metrics import confusion_matrix, classification_report, roc_auc_score

# Load mixed dataset (both normal and DDoS)
mixed_data = pd.read_csv("mixed_traffic_dataset.csv")

# Extract labels (if present)
labels = mixed_data['label'] if 'label' in mixed_data.columns else None

# Drop label column from features
if 'label' in mixed_data.columns:
    mixed_data = mixed_data.drop(columns=['label'])

# Load encoder and scaler
encoder = load("encoder_iso.joblib")
scaler = load("scaler_iso.joblib")

# Encode categorical features
cat_cols = ['Protocol']
encoded = encoder.transform(mixed_data[cat_cols])
encoded_df = pd.DataFrame(encoded, columns=encoder.get_feature_names_out(cat_cols))

```

Figure 16 Isolation Forest model evaluation

```

# Combine with numeric features
num_cols = mixed_data.select_dtypes(include=[np.number]).columns
X_mixed = pd.concat([mixed_data[num_cols].reset_index(drop=True), encoded_df.reset_index(drop=True)], axis=1)

# Fill missing values
X_mixed = X_mixed.fillna(0)

# Scale features
X_scaled = scaler.transform(X_mixed)

# Load trained model
model = load("isolation_forest_ddos_model.joblib")

# Predict anomalies
predictions = model.predict(X_scaled)
predictions = np.where(predictions == -1, 1, 0) # 1 = anomaly (DDoS), 0 = normal

# Evaluate if labels exist
if labels is not None:
    cm = confusion_matrix(labels, predictions)
    print("Confusion Matrix:\n", cm)
    print("\nClassification Report:\n", classification_report(labels, predictions))
    print("ROC AUC:", roc_auc_score(labels, predictions))
else:
    print("No labels found in dataset. Only predictions generated.")

```

Figure 17 Isolation Forest model evaluation

```

else:
    print("No labels found in dataset. Only predictions generated.")

# Save predictions
output = pd.DataFrame(X_mixed)
output['predicted_label'] = predictions
if labels is not None:
    output['true_label'] = labels.values
output.to_csv("mixed_predictions.csv", index=False)

print("Predictions saved as 'mixed_predictions.csv'.")

```

[2]

```

... Confusion Matrix:
      [[33250  1750]
       [   470 35322]]

```

Classification Report:

	precision	recall	f1-score	support
0	0.99	0.95	0.97	35000
1	0.95	0.99	0.97	35792
accuracy			0.97	70792
macro avg	0.97	0.97	0.97	70792
weighted avg	0.97	0.97	0.97	70792

Figure 18 Isolation Forest model evaluation

Isolation Forest Model Integration and Detection Codes on Home Assistant with Flask API

```
live_ddos_detection.py > on_message
1  import paho.mqtt.client as mqtt
2  import requests
3  import json
4  import time
5  from dotenv import load_dotenv
6  import os
7
8  # --- CONFIGURATION ---
9  API_URL = "http://192.168.0.61:5000/predict"
10 MQTT_BROKER = "192.168.254.129"
11 MQTT_PORT = 1883
12 MQTT_USER = "mqtt_user"
13 MQTT_PASS = "Dissertation@1"
14 MQTT_TOPIC_DETECTION = "home/traffic/ddos_alert"
15 MQTT_TOPICS = [
16     "home/sensor/temperature",
17     "home/sensor/humidity",
18     "home/sensor/gas"
19 ]
20
21 load_dotenv() # loads the .env file
22 HA_URL = os.getenv("HA_URL")
23 HA_TOKEN = os.getenv("HA_TOKEN")
24
25 HEADERS = {
26     "Authorization": f"Bearer {HA_TOKEN}",
27     "Content-Type": "application/json",
28 }
```

Figure 19 Isolation Forest Integration using Flask API and Detection

```

# --- STATE ---
latest = {"temperature": None, "humidity": None, "gas": None}
packet_counter = 0
last_time = None

# --- HELPER FUNCTIONS ---
def fetch_ha_sensors():
    """Optional: Fetch current HA sensor states."""
    try:
        response = requests.get(f"{HA_URL}/api/states", headers=HEADERS, timeout=3)
        response.raise_for_status()
        sensors = response.json()
        for sensor in sensors:
            entity_id = sensor["entity_id"]
            state = sensor["state"]
            # Map Home Assistant entities to latest dictionary if names match
            if "temperature" in entity_id:
                latest["temperature"] = float(state)
            elif "humidity" in entity_id:
                latest["humidity"] = float(state)
            elif "gas" in entity_id:
                latest["gas"] = float(state)
    except Exception as e:
        print(f"Failed to fetch HA sensors: {e}")

# --- MQTT CALLBACK ---
def on_message(client, userdata, msg):
    global packet_counter, last_time

```

Figure 20 Isolation Forest Integration using Flask API and Detection

```

58     topic = msg.topic
59     value = float(msg.payload.decode())
60
61     if "temperature" in topic:
62         latest["temperature"] = value
63     elif "humidity" in topic:
64         latest["humidity"] = value
65     elif "gas" in topic:
66         latest["gas"] = value
67
68     # Only run detection if all sensors have a value
69     if all(v is not None for v in latest.values()):
70         now = time.time()
71         inter_arrival = round(now - last_time, 6) if last_time else 0.0
72         last_time = now
73         packet_counter += 1
74
75         # Prepare features for your model
76         features = {
77             "no": packet_counter,
78             "time": now,
79             "temperature": latest["temperature"],
80             "humidity": latest["humidity"],
81             "length": latest["gas"],
82             "protocol": "MQTT"
83         }
84
85         # Call API

```

Figure 21 Isolation Forest Integration using Flask API and Detection

```

86         try:
87             resp = requests.post(API_URL, json=features, timeout=3)
88             resp.raise_for_status()
89             result = resp.json()
90         except Exception as e:
91             print(f"API error: {e}")
92             result = {"prediction": "Unknown", "score": None}
93
94         # Print result
95         print(f"Packet {packet_counter}: Temp={latest['temperature']} "
96               f"Humidity={latest['humidity']} Gas={latest['gas']} => "
97               f"Detection: {result.get('prediction')} Score: {result.get('score')}")
98
99         # Publish to MQTT
100         label_raw = result.get("prediction")
101         label = "DDOS" if isinstance(label_raw, str) and "ddos" in label_raw.lower() else "NORMAL"
102         out = {
103             "label": label,
104             "score": result.get("score"),
105             "model_raw": result.get("raw_prediction"),
106             "meta": {
107                 "packet_no": packet_counter,
108                 "temperature": latest["temperature"],
109                 "humidity": latest["humidity"],
110                 "packet_length": latest["gas"],
111                 "inter_arrival": inter_arrival
112             },
113             "timestamp": time.strftime("%Y-%m-%dT%H:%M:%SZ", time.gmtime())

```

Figure 22 Isolation Forest Integration using Flask API and Detection

```

113         "timestamp": time.strftime("%Y-%m-%dT%H:%M:%SZ", time.gmtime())
114     }
115     retain = True if label == "DDOS" else False
116     client.publish(MQTT_TOPIC_DETECTION, json.dumps(out), retain=retain)
117
118     # --- MQTT SETUP ---
119     mqtt_client = mqtt.Client()
120     mqtt_client.username_pw_set(MQTT_USER, MQTT_PASS)
121     mqtt_client.on_message = on_message
122     mqtt_client.connect(MQTT_BROKER, MQTT_PORT, 60)
123     for t in MQTT_TOPICS:
124         mqtt_client.subscribe(t)
125
126     print("Listening for sensor data and sending live detection results...")
127     try:
128         mqtt_client.loop_forever()
129     except KeyboardInterrupt:
130         print("Stopped.")
131         mqtt_client.disconnect()

```

Figure 23 Isolation Forest Integration using Flask API and Detection

XGBoost Model building codes with evaluation

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
from xgboost import XGBClassifier
from joblib import dump

# Load CSVs
ddos_data = pd.read_csv("DDoS traffic.csv")
normal_data = pd.read_csv("normal traffic.csv")

normal_data['label'] = 0
ddos_data['label'] = 1

df = pd.concat([normal_data, ddos_data], ignore_index=True)
print("Loaded dataset:", df.shape)

# Clean data
# Drop completely empty columns
df = df.dropna(axis=1, how='all')

# Fill missing numeric values
for col in df.select_dtypes(include=[np.number]).columns:
    df[col] = df[col].fillna(0)
```

Figure 24 XGBoost model building and evaluation

```
# Encode non-numeric columns
for col in df.select_dtypes(exclude=[np.number]).columns:
    df[col] = df[col].astype(str)
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])

# Separate features and label
X = df.drop(columns=['label'])
y = df['label']

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# Scale numeric features
scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s = scaler.transform(X_test)

# XGBoost model (handles imbalance well)
model = XGBClassifier(
    n_estimators=400,
```

Figure 25 XGBoost model building and evaluation

```

n_estimators=400,
learning_rate=0.1,
max_depth=8,
subsample=0.8,
colsample_bytree=0.8,
scale_pos_weight=(len(y_train[y_train==0]) / len(y_train[y_train==1])),
random_state=42,
n_jobs=-1,
eval_metric='logloss'
)

model.fit(X_train_s, y_train)

# Evaluate
y_pred = model.predict(X_test_s)
y_prob = model.predict_proba(X_test_s)[:,-1]

print("\n=== Classification Report ===")
print(classification_report(y_test, y_pred, digits=4))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

try:
    print("ROC AUC:", round(roc_auc_score(y_test, y_prob), 4))
except:

```

Figure 26 XGBoost model building and evaluation

```

pass

# Save model + scaler
dump(model, "Xgboost_ddos_model.joblib")
dump(list(X.columns), "columns_xgboost.joblib")
dump(scaler, "scaler_xgboost.joblib")

print("\nModel saved as Xgboost_ddos_model.joblib")

```

Loaded dataset: (70792, 8)

```

=== Classification Report ===

```

	precision	recall	f1-score	support
0	1.0000	0.9999	0.9999	7000
1	0.9999	1.0000	0.9999	7159
accuracy			0.9999	14159
macro avg	0.9999	0.9999	0.9999	14159
weighted avg	0.9999	0.9999	0.9999	14159

```

Confusion Matrix:
[[6999  1]

```

Figure 27 XGBoost model building and evaluation

XGBoost Model integration and detection with Home Assistant using Flask API

```
model_API.py > ...
1  from flask import Flask, request, jsonify
2
3  app = Flask(__name__)
4
5  @app.route("/predict", methods=["POST"])
6  def predict():
7      data = request.json
8      # Dummy prediction for testing
9      return jsonify({"prediction": "NORMAL", "score": 0.9, "raw_prediction": data})
10
11 if __name__ == "__main__":
12     # Listen on all network interfaces so other machines can reach it
13     app.run(host="0.0.0.0", port=5000, debug=True)
```

Figure 28 XGBoost model integration with Home Assistant and live Detection

```
live_ddos_detection.py > ...
1  import paho.mqtt.client as mqtt
2  import requests
3  import json
4  import time
5  from dotenv import load_dotenv
6  import os
7
8  # --- CONFIGURATION ---
9  API_URL = "http://127.0.0.1:5000/predict"
10 MQTT_BROKER = "192.168.254.129"
11 MQTT_PORT = 1883
12 MQTT_USER = "mqtt_user"
13 MQTT_PASS = "Dissertation@1"
14
15 MQTT_TOPIC_DETECTION = "home/traffic/ddos_alert"
16 MQTT_TOPICS = [
17     "home/sensor/temperature",
18     "home/sensor/humidity",
19     "home/sensor/gas"
20 ]
21
22 load_dotenv()
23 HA_URL = os.getenv("HA_URL")
24 HA_TOKEN = os.getenv("HA_TOKEN")
25
26 HEADERS = {
27     "Authorization": f"Bearer {HA_TOKEN}",
28     "Content-Type": "application/json",
```

Figure 29 XGBoost model integration with Home Assistant and live Detection

```

29     }
30
31     # --- STATE ---
32     latest = {
33         "temperature": None,
34         "humidity": None,
35         "gas": None
36     }
37
38     packet_counter = 0
39     last_time = None
40
41     # --- MQTT CALLBACK ---
42     def on_message(client, userdata, msg):
43         global packet_counter, last_time
44
45         topic = msg.topic
46         value = float(msg.payload.decode())
47
48         if "temperature" in topic:
49             latest["temperature"] = value
50         elif "humidity" in topic:
51             latest["humidity"] = value
52         elif "gas" in topic:
53             latest["gas"] = value
54
55         # Run prediction only when all values exist

```

Figure 30 XGBoost model integration with Home Assistant and live Detection

```

56     if not all(v is not None for v in latest.values()):
57         return
58
59     now = time.time()
60     inter_arrival = round(now - last_time, 6) if last_time else 0.0
61     last_time = now
62     packet_counter += 1
63
64     # --- XGBOOST FEATURE VECTOR ---
65     features = {
66         "packet_no": packet_counter,
67         "temperature": latest["temperature"],
68         "humidity": latest["humidity"],
69         "packet_length": latest["gas"],
70         "inter_arrival": inter_arrival
71     }
72
73     # --- API CALL ---
74     try:
75         response = requests.post(API_URL, json=features, timeout=3)
76         response.raise_for_status()
77         result = response.json()
78     except Exception as e:
79         print(f"API error: {e}")
80         return
81
82     # --- MODEL OUTPUT ---

```

Figure 31 XGBoost model integration with Home Assistant and live Detection

```

83     prediction = result.get("prediction")          # 0 or 1
84     probability = result.get("probability")        # float
85
86     label = "DDOS" if prediction == 1 else "NORMAL"
87
88     print(
89         f"Packet {packet_counter} | "
90         f"Temp={latest['temperature']} "
91         f"Humidity={latest['humidity']} "
92         f"Length={latest['gas']} | "
93         f"Prediction={label} "
94         f"Prob={probability}"
95     )
96
97     # --- MQTT PUBLISH ---
98     payload = {
99         "label": label,
100         "probability": probability,
101         "features": features,
102         "timestamp": time.strftime("%Y-%m-%dT%H:%M:%SZ", time.gmtime())
103     }
104
105     retain = True if label == "DDOS" else False
106     client.publish(MQTT_TOPIC_DETECTION, json.dumps(payload), retain=retain)
107
108     # --- MQTT SETUP ---
109     mqtt_client = mqtt.Client()

```

Figure 32 XGBoost model integration with Home Assistant and live Detection

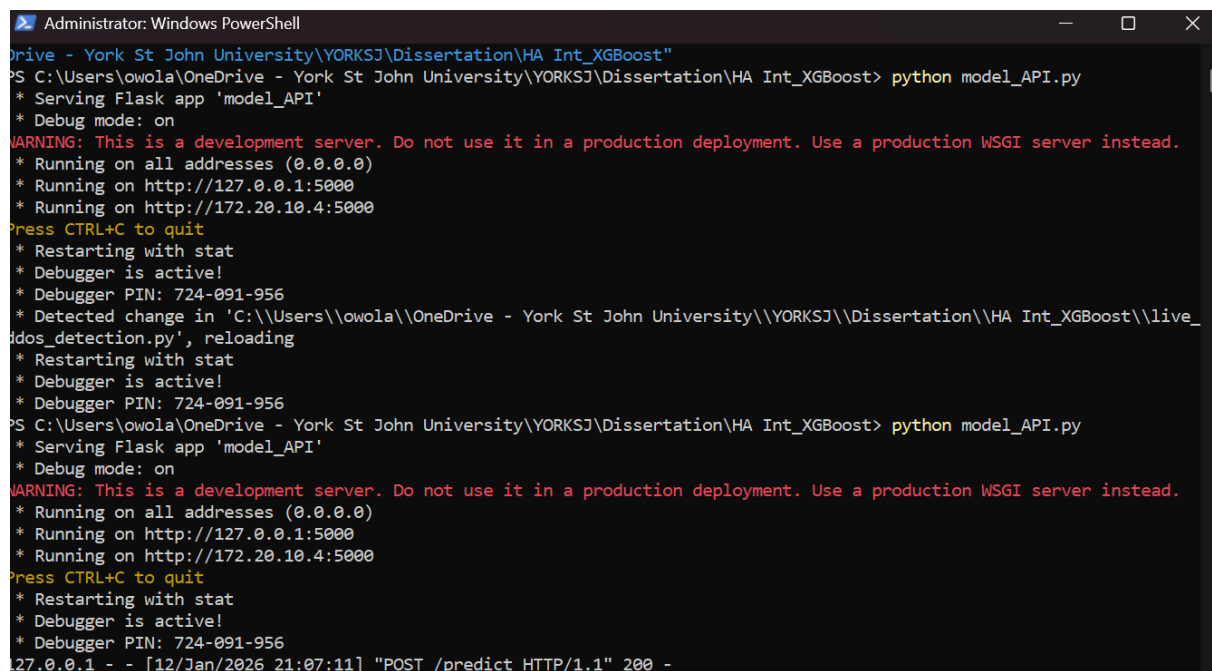
```

109     mqtt_client = mqtt.Client()
110     mqtt_client.username_pw_set(MQTT_USER, MQTT_PASS)
111     mqtt_client.on_message = on_message
112
113     mqtt_client.connect(MQTT_BROKER, MQTT_PORT, 60)
114
115     for topic in MQTT_TOPICS:
116         mqtt_client.subscribe(topic)
117
118     print("Listening for sensor data and sending XGBoost DDoS predictions...")
119
120     try:
121         mqtt_client.loop_forever()
122     except KeyboardInterrupt:
123         mqtt_client.disconnect()
124         print("Stopped.")
125

```

Figure 33 XGBoost model integration with Home Assistant and live Detection

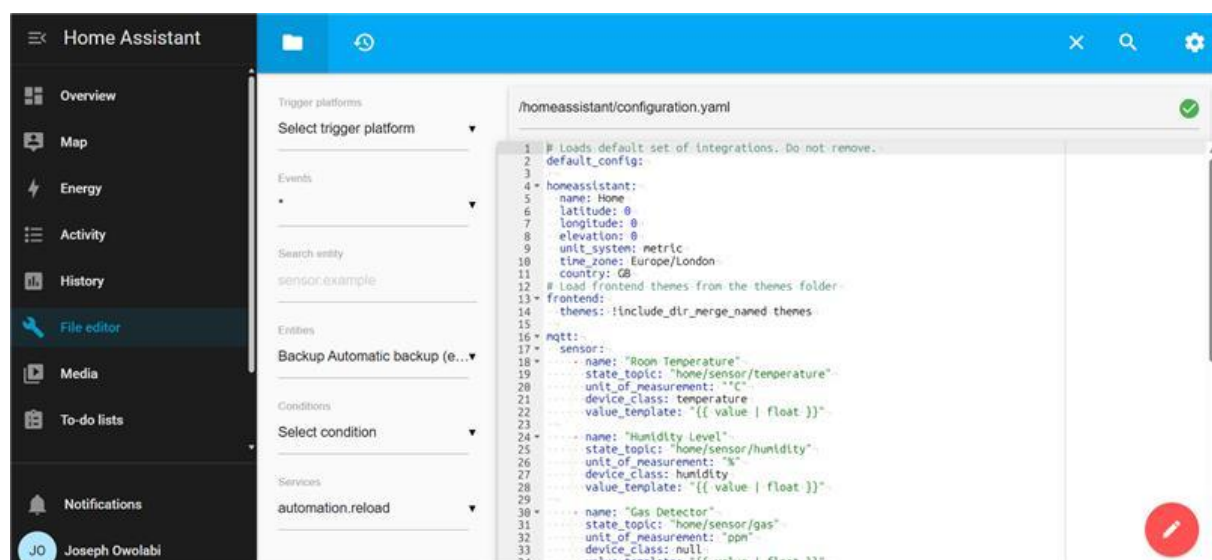
Flask API running on PowerShell for XGBoost



```
Administrator: Windows PowerShell
PS C:\Users\owola\OneDrive - York St John University\YORKSJ\Dissertation\HA Int_XGBoost> python model_API.py
* Serving Flask app 'model_API'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.20.10.4:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 724-091-956
* Detected change in 'C:\Users\owola\OneDrive - York St John University\YORKSJ\Dissertation\HA Int_XGBoost\live_
dos_detection.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 724-091-956
PS C:\Users\owola\OneDrive - York St John University\YORKSJ\Dissertation\HA Int_XGBoost> python model_API.py
* Serving Flask app 'model_API'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.20.10.4:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 724-091-956
27.0.0.1 - - [12/Jan/2026 21:07:11] "POST /predict HTTP/1.1" 200 -
```

Figure 34 Flask API on Powershell

Home Assistant YAML Configuration Script



```
/homeassistant/configuration.yaml
1 # Loads default set of integrations. Do not remove.
2 default_config:
3
4 - homeassistant:
5   name: Home
6   latitude: 0
7   longitude: 0
8   elevation: 0
9   unit_system: metric
10  time_zone: Europe/London
11  country: GB
12
13 # Load frontend themes from the themes folder
14 frontend:
15   themes: !include_dir_merge_named themes
16
17 mqtt:
18   sensor:
19     - name: "Room Temperature"
20       state_topic: "home/sensor/temperature"
21       unit_of_measurement: "°C"
22       device_class: temperature
23       value_template: "{{ value | float }}"
24
25     - name: "Humidity Level"
26       state_topic: "home/sensor/humidity"
27       unit_of_measurement: "%"
28       device_class: humidity
29       value_template: "{{ value | float }}"
30
31     - name: "Gas Detector"
32       state_topic: "home/sensor/gas"
33       unit_of_measurement: "ppm"
34       device_class: null
35       value_template: "{{ value | float }}"
```

Figure 35 Home Assistant Yaml Configuration

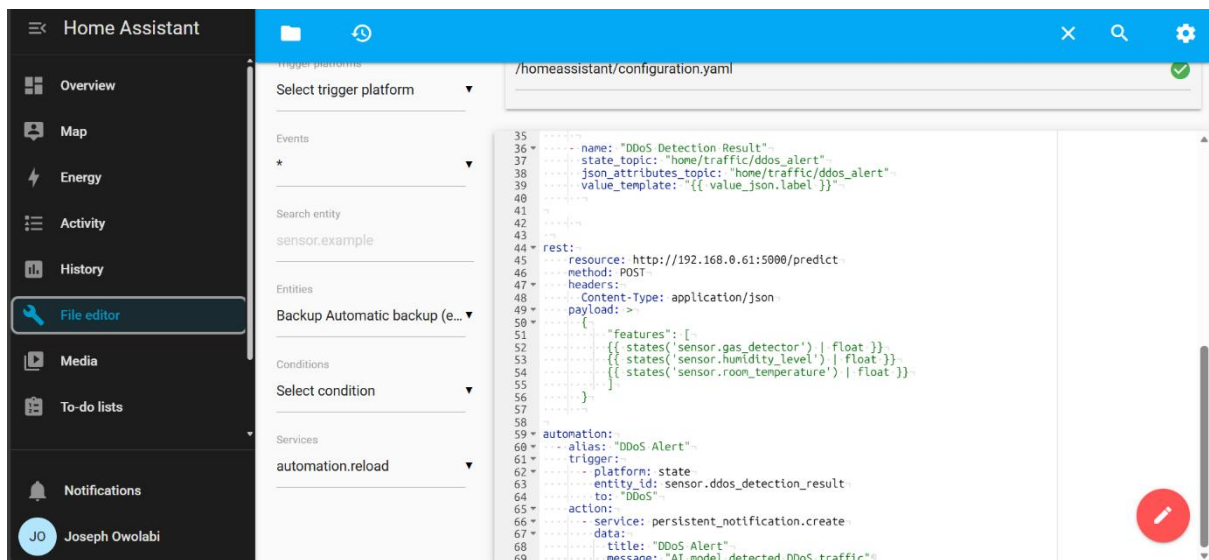


Figure 36 Home Assistant Yaml Configuration

Normal Traffic Network Capture

Normal traffic capture.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

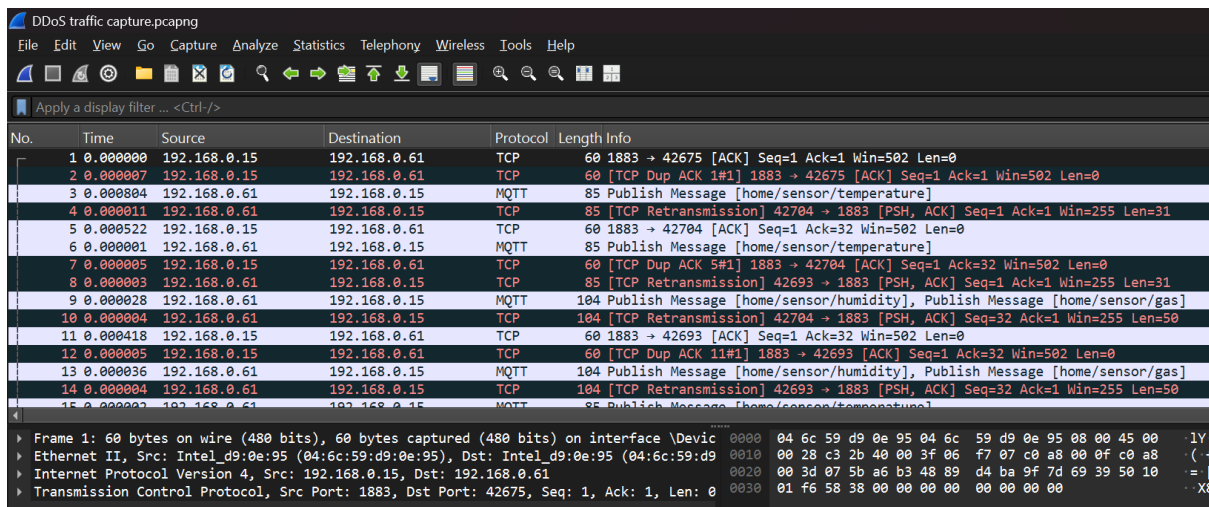
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.61	192.168.0.15	TCP	66	53007 → 8123 [PSH, ACK] Seq=1 Ack=1 Win=253 Len=12
2	0.000009	192.168.0.61	192.168.0.15	TCP	66	[TCP Retransmission] 53007 → 8123 [PSH, ACK] Seq=1 Ack=1 Win=253 Len=12
3	0.001173	192.168.0.15	192.168.0.61	TCP	175	8123 → 53007 [PSH, ACK] Seq=1 Ack=13 Win=501 Len=121
4	0.000006	192.168.0.15	192.168.0.61	TCP	175	[TCP Retransmission] 8123 → 53007 [PSH, ACK] Seq=1 Ack=13 Win=501 Len=121
5	0.050388	192.168.0.61	192.168.0.15	TCP	54	53007 → 8123 [ACK] Seq=13 Ack=122 Win=253 Len=0
6	0.000027	192.168.0.61	192.168.0.15	TCP	54	[TCP Dup ACK 5#1] 53007 → 8123 [ACK] Seq=13 Ack=122 Win=253 Len=0
7	0.712252	192.168.0.98	192.168.0.15	UDP	220	64041 → 38900 Len=178
8	4.323147	192.168.0.15	192.168.0.61	TCP	232	8123 → 53007 [PSH, ACK] Seq=122 Ack=13 Win=501 Len=178
9	0.000016	192.168.0.15	192.168.0.61	TCP	232	[TCP Retransmission] 8123 → 53007 [PSH, ACK] Seq=122 Ack=13 Win=501 Len=178
10	0.049766	192.168.0.61	192.168.0.15	TCP	54	53007 → 8123 [ACK] Seq=13 Ack=300 Win=252 Len=0
11	0.000025	192.168.0.61	192.168.0.15	TCP	54	[TCP Dup ACK 10#1] 53007 → 8123 [ACK] Seq=13 Ack=300 Win=252 Len=0
12	0.375715	192.168.0.98	192.168.0.15	UDP	220	64041 → 38900 Len=178
13	1.746206	192.168.0.61	192.168.0.15	MQTT	85	Publish Message [home/sensor/temperature]
14	0.000017	192.168.0.61	192.168.0.15	TCP	85	[TCP Retransmission] 24760 → 1883 [PSH, ACK] Seq=1 Ack=1 Win=255 Len=31
15	0.000557	192.168.0.15	192.168.0.61	TCP	60	1883 → 24760 [ACK] Seq=1 Ack=33 Win=501 Len=0

Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device...
 Ethernet II, Src: Intel_d9:0e:95 (04:6c:59:d9:0e:95), Dst: Intel_d9:0e:95 (04:6c:59:d9...
 Internet Protocol Version 4, Src: 192.168.0.61, Dst: 192.168.0.15
 Transmission Control Protocol, Src Port: 53007, Dst Port: 8123, Seq: 1, Ack: 1, Len: 1
 Data (12 bytes)

Figure 37 Normal Traffic capture

DDoS Traffic Network Capture

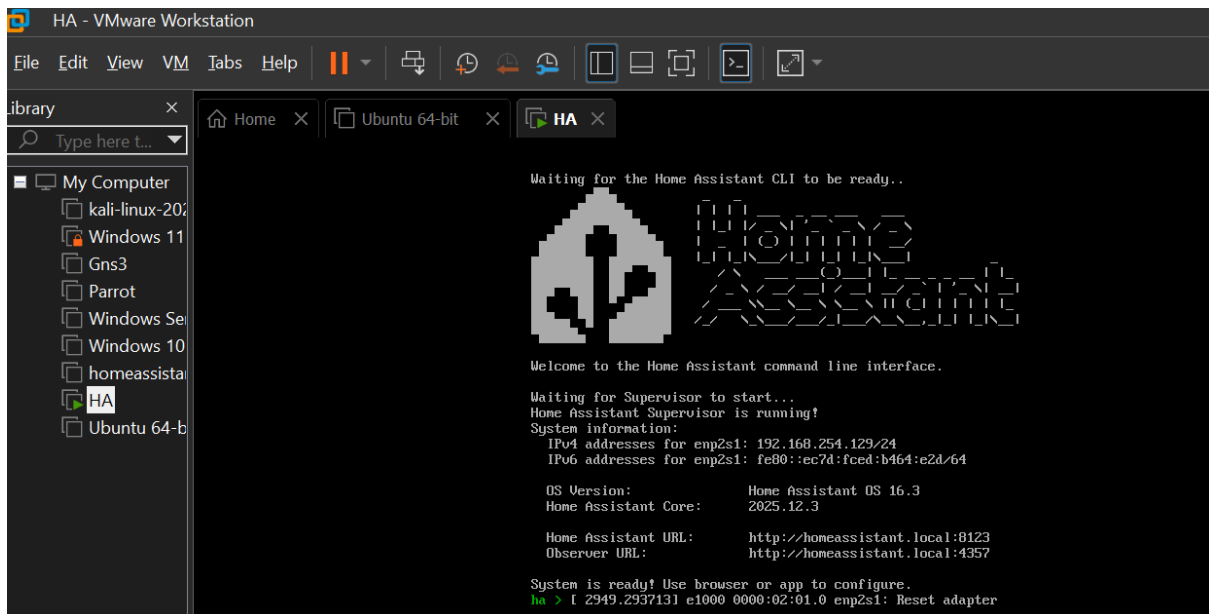


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.15	192.168.0.61	TCP	60	1883 → 42675 [ACK] Seq=1 Ack=1 Win=502 Len=0
2	0.000007	192.168.0.15	192.168.0.61	TCP	60	[TCP Dup ACK 1#1] 1883 → 42675 [ACK] Seq=1 Ack=1 Win=502 Len=0
3	0.000004	192.168.0.61	192.168.0.15	MQTT	85	Publish Message [home/sensor/temperature]
4	0.000011	192.168.0.61	192.168.0.15	TCP	85	[TCP Retransmission] 42704 → 1883 [PSH, ACK] Seq=1 Ack=1 Win=255 Len=31
5	0.000522	192.168.0.15	192.168.0.61	TCP	60	1883 → 42704 [ACK] Seq=1 Ack=32 Win=502 Len=0
6	0.000001	192.168.0.61	192.168.0.15	MQTT	85	Publish Message [home/sensor/temperature]
7	0.000005	192.168.0.15	192.168.0.61	TCP	60	[TCP Dup ACK 5#1] 1883 → 42704 [ACK] Seq=1 Ack=32 Win=502 Len=0
8	0.000003	192.168.0.61	192.168.0.15	TCP	85	[TCP Retransmission] 42693 → 1883 [PSH, ACK] Seq=1 Ack=1 Win=255 Len=31
9	0.000028	192.168.0.61	192.168.0.15	MQTT	104	Publish Message [home/sensor/humidity], Publish Message [home/sensor/gas]
10	0.000004	192.168.0.61	192.168.0.15	TCP	104	[TCP Retransmission] 42704 → 1883 [PSH, ACK] Seq=32 Ack=1 Win=255 Len=50
11	0.000418	192.168.0.15	192.168.0.61	TCP	60	1883 → 42693 [ACK] Seq=1 Ack=32 Win=502 Len=0
12	0.000005	192.168.0.15	192.168.0.61	TCP	60	[TCP Dup ACK 11#1] 1883 → 42693 [ACK] Seq=1 Ack=32 Win=502 Len=0
13	0.000036	192.168.0.61	192.168.0.15	MQTT	104	Publish Message [home/sensor/humidity], Publish Message [home/sensor/gas]
14	0.000004	192.168.0.61	192.168.0.15	TCP	104	[TCP Retransmission] 42693 → 1883 [PSH, ACK] Seq=32 Ack=1 Win=255 Len=50
15	0.000003	192.168.0.61	192.168.0.15	MQTT	85	Publish Message [home/sensor/temperature]

Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface Device
 Ethernet II, Src: Intel_d9:0e:95 (04:6c:59:d9:0e:95), Dst: Intel_d9:0e:95 (04:6c:59:d9:0e:95)
 Internet Protocol Version 4, Src: 192.168.0.15, Dst: 192.168.0.61
 Transmission Control Protocol, Src Port: 1883, Dst Port: 42675, Seq: 1, Ack: 1, Len: 0

Figure 38 DDoS Traffic capture

Home Assistant Command Line Interface (CLI)



```

HA - VMware Workstation
File Edit View VM Tabs Help
library
Type here to...
My Computer
kali-linux-20...
Windows 11
Gns3
Parrot
Windows Se...
Windows 10
homeassista...
HA
Ubuntu 64-b...

Waiting for the Home Assistant CLI to be ready..

Home Assistant
Assistant
Command Line
Interface

Welcome to the Home Assistant command line interface.

Waiting for Supervisor to start...
Home Assistant Supervisor is running!
System information:
IPv4 addresses for emp2s1: 192.168.254.129/24
IPv6 addresses for emp2s1: fe80::ec7d:fcd:b464:e2d/64

OS Version: Home Assistant OS 16.3
Home Assistant Core: 2025.12.3

Home Assistant URL: http://homeassistant.local:8123
Observer URL: http://homeassistant.local:4357

System is ready! Use browser or app to configure.
ha > I 2949.2937131 e1000 0000:02:01:0 emp2s1: Reset adapter
  
```

Figure 39 Home Assistant CLI

Home Assistant Integrations Interface

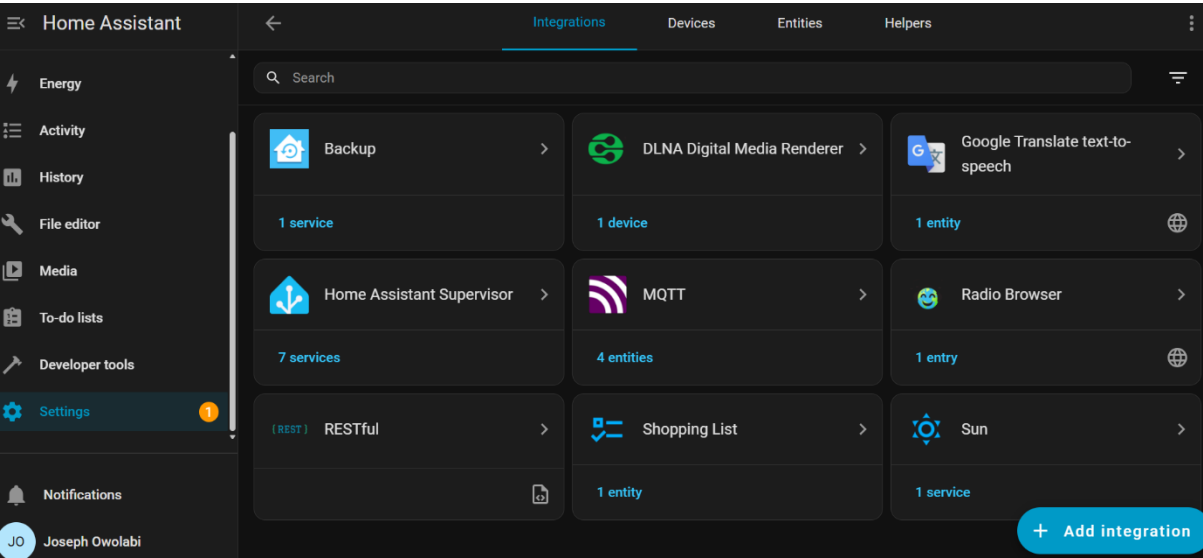


Figure 40 Home Assistant Integrations