University of Central Lancashire
UCLan

Central Lancashire Online Knowledge (CLoK)

Title	SDN-based detection and mitigation of DDoS attacks on smart homes
Type	Article
URL	https://clok.uclan.ac.uk/id/eprint/51129/
DOI	https://doi.org/10.1016/j.comcom.2024.04.001
Date	2024
Citation	Garba, Usman Haruna, Toosi, Adel N., Pasha, Muhammad Fermi and Khan, Suleman (2024) SDN-based detection and mitigation of DDoS attacks on smart homes. Computer Communications, 221. pp. 29-41. ISSN 0140-3664
Creators	Garba, Usman Haruna, Toosi, Adel N., Pasha, Muhammad Fermi and Khan, Suleman

It is advisable to refer to the publisher's version if you intend to cite from the work.
https://doi.org/10.1016/j.comcom.2024.04.001

For information about Research at UCLan please go to http://www.uclan.ac.uk/research/

All outputs in CLoK are protected by Intellectual Property Rights law, including Copyright law. Copyright, IPR and Moral Rights for the works on this site are retained by the individual authors and/or other copyright owners. Terms and conditions for use of this material are defined in the http://clok.uclan.ac.uk/policies/

Computer Communications 221 (2024) 29–41

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

SDN-based detection and mitigation of DDoS attacks on smart homes☆

Usman Haruna Garba [a], Adel N. Toosi [b,*], Muhammad Fermi Pasha [a], Suleman Khan [c]

[a] School of Information Technology, Monash University, Selangor, Malaysia
[b] Faculty of Information Technology, Monash University, Clayton VIC 3800, Australia
[c] School of Psychology and Computer Science, University of Central Lancashire, Newcastle, United Kingdom

ARTICLE INFO

Keywords:
Software-defined networks
Internet of Things
DDoS attacks
Intrusion detection system
Supervised classification

ABSTRACT

The adoption of the Internet of Things (IoT) has proliferated across various domains, where everyday objects like refrigerators and washing machines are now equipped with sensors and connected to the internet. Undeniably, the security of such devices, which were not primarily designed for internet connectivity, is of utmost importance but has been largely neglected. In this paper, we propose a framework for the real-time DDoS attack detection and mitigation in SDN-enabled smart home networks. We capture network traffic during regular operations and during DDoS attacks. This captured traffic is used to train several machine learning (ML) models, including Support Vector Machine (SVM), Logistic Regression, Decision Trees, and K-Nearest Neighbors (KNN) algorithms. These trained models are executed as SDN controller applications and subsequently employed for real-time attack detection. While we utilize ML techniques to protect IoT devices, we propose the use of SNORT, a signature-based detection technique, to secure the SDN controller itself. Real-world experiments demonstrate that without SNORT, the SDN controller goes offline shortly after an attack, resulting in a 100% packet loss. Furthermore, we show that ML algorithms can efficiently classify traffic into benign and attack traffic, with the Decision Tree algorithm outperforming others with an accuracy of 99%.

1. Introduction

In recent years, the proliferation of smart home technologies has significantly enhanced residential convenience and efficiency. However, this advancement has also introduced new vulnerabilities, particularly to Distributed Denial of Service (DDoS) attacks. Current methodologies in network security, while effective to a degree, often struggle with the real-time detection and mitigation of such attacks in increasingly complex and interconnected home networks.

This paper addresses these critical challenges by proposing a novel framework that leverages traditional Machine Learning (ML) techniques, tailored specifically for the nuanced environment of smart home networks. Our approach is designed to not only detect but also mitigate DDoS attacks in real-time, bridging a crucial gap in current security methodologies.

The relevance of our approach is underscored by its alignment with the latest trends in network security and IoT device integration. By focusing on real-time processing and adapting ML models to the specific context of smart home networks, our methodology presents a timely and effective solution to an escalating security concern.

In a typical smart home, various sensors (such as temperature, humidity, and pressure sensors) and household items like refrigerators, washing machines, IP cameras, smart bulbs, and alarm systems are connected to the internet. They communicate with remote applications and servers to collect, share, and transmit data from the environment [1], aiming to simplify the homeowner's life [2]. However, due to the limited processing capability and memory in these devices, they often lack embedded security mechanisms, becoming vulnerable launchpads for attacks against different systems. As reported by [3], a DDoS attack on smart TVs and Blu-ray DVDs can result in these devices entering an endless reboot loop, rendering them unusable for the homeowner.

Information and communication resources in a smart home environment are crucial to homeowners; thus, the security of the smart home is paramount, yet it can be compromised by DDoS attacks. For instance, encrypting user credentials in a smart refrigerator is impractical due to limited bandwidth, memory, and processing power required for encryption and decryption tasks [4]. In 2016, vulnerable IoT devices were

☆ This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.
* Corresponding author.
E-mail addresses: usman.garba@monash.edu (U.H. Garba), adel.n.toosi@monash.edu (A.N. Toosi), muhammad.fermipasha@monash.edu (M.F. Pasha), suleman.khan@northumbria.ac.uk (S. Khan).
URLs: https://usmangarba.com (U.H. Garba), https://adelnadjarantoosi.info (A.N. Toosi), https://www.monash.edu.my/it/staff/academic/dr-muhammad-fermi-pasha (M.F. Pasha), https://www.uclan.ac.uk/academics/suleman-khan (S. Khan).

https://doi.org/10.1016/j.comcom.2024.04.001
Received 31 October 2023; Received in revised form 8 March 2024; Accepted 1 April 2024
Available online 3 April 2024
0140-3664/© 2024 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

exploited to carry out one of the largest Distributed Denial of Service (DDoS) attacks using malware known as the Mirai IoT Botnet [5]. The Mirai IoT botnet compromised over 150,000 IoT devices, subsequently launching a DDoS attack against a DNS provider, DYN [6].

Software-defined network (SDN) architecture provides logically centralized network control, programmable networks, and network abstraction [7]. In this paper, we propose a real-time DDoS attack detection and mitigation framework in an SDN-enabled smart home network. The approach combines a Signature-based Intrusion Detection System (IDS) and ML algorithms to detect such attacks from malicious hosts within the smart home network in real-time without interrupting network operation between devices in the Smart home.

The SDN controller, central to the IoT testbed and our experiments, controls the IoT devices and directs traffic between each device, and the proposed detection algorithms are deployed on the SDN controller as an application. We use the RYU SDN Controller, which is responsible for the entire network's operation in the smart home, to connect various IoT devices. The network traffic between these devices is captured while TCP SYN flood DDoS attacks are conducted from several hosts in the network, targeting the SDN controller and IoT devices. Hence, the controller is crucial to the success of the proposed detection technique.

The SDN controller itself is susceptible to DDoS attacks with devastating effects on the operation of the IoT testbed, as shown in our experiments. To protect the SDN controller from such attacks, we deploy SNORT IDS with rules in its database to match the controller's IP address and Port number. SNORT IDS is suitable for detecting known attacks by using a database of signatures from known attacks [2]. However, it is not suitable for IoT network traffic due to the heterogeneous characteristics of IoT devices and the daunting task of creating rules for every IoT device in the smart home network. Therefore, trained ML models are deployed on the SDN controller to protect IoT devices from DDoS attacks in the network in real-time. In summary, the **main contributions** of this work are as follows:

1. We propose a real-time DDoS attack detection and mitigation approach, enabling legitimate IoT devices to continue communicating in a smart home with minimal network interruption. This approach is scalable as it can be deployed on the SDN controller without incurring extra deployment costs in large-scale networks.
2. Using dimension reduction techniques, we identify and select an accurate set of features that maximize relevance, reduce redundancy, and decrease the number of features, thereby enhancing the overall detection accuracy of the proposed framework.
3. We conduct an evaluation of the proposed system in a real-world testbed and perform a comprehensive performance comparison of the results using a DDoS attack dataset available in the public domain alongside the dataset captured from the IoT testbed. The results from the evaluation and comparisons with standard datasets indicate that the proposed technique is scalable and improves detection accuracy while also minimizing false positives.

The remainder of the paper is structured as follows. In Section 2, We introduce the background of the study and discuss different types of DDoS attacks, key considerations for detecting and mitigating DDoS attacks, along with privacy and security considerations. Section 3 presents the proposed frameworks, covering the SNORT IDS framework and the Detection Architecture, which includes the different components of the proposed real-time detection technique. In Section 4, we identify the selected ML algorithms used alongside the proposed framework for real-time detection, Feature Extraction techniques, network topology, and finally the attack scenarios. In Section 5, we conduct the performance evaluation of the detection techniques with standard datasets, compare the results, and discuss the outcomes of the proposed framework. Next, we discuss the role of our techniques in enhancing DDoS detection in SDN-enabled smart homes and propose future research directions in Section 6. Section 7 presents related works. Finally, Section 8 concludes the paper and highlights the open research areas and concludes the work.

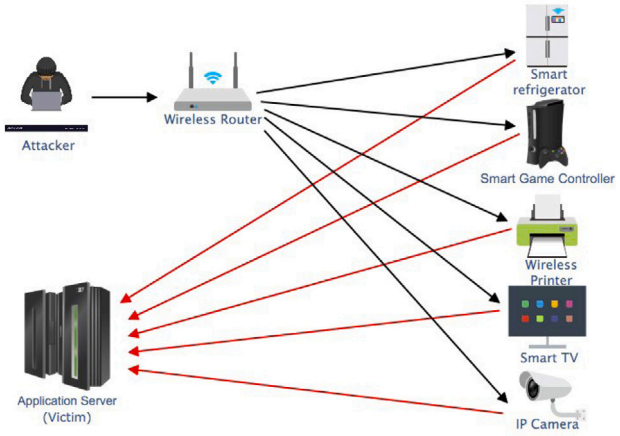

Fig. 1. Reflective DDoS attack using IoT devices.

2. Background

In this section, we explore different types of attacks, essential factors for DDoS detection and mitigation in IoT networks, and considerations for privacy and security.

2.1. Various types of attacks

2.1.1. UDP flood attack

A User Datagram Protocol (UDP) flood attack is a type of DDoS attack that floods a victim device with random UDP requests. The victim's device continuously monitors the listening port and responds with an unreachable destination message (ICMP) since no application utilizes those random port numbers. Consequently, this process depletes the victim's resources, rendering it incapable of responding to legitimate UDP packets [8].

2.1.2. TCP syn attack

In TCP SYN attacks, a component of DDoS tactics, the attacker leverages the TCP connection procedure. The attacker initiates a SYN request to the server, prompting a SYN-ACK response, but then fails to respond or redirects the response, resulting in the server being left with open, incomplete connections [9]. In other words, the attacker can utilize vulnerable IoT devices to flood the victim with SYN requests. The attacker then either fails to respond to the SYN-ACK or redirects the response to a spoofed IP address of another victim. This leaves the server with open ports awaiting SYN-ACK responses, resulting in multiple open connections without responses. As a consequence, the server's resources are depleted, causing it to deny responses to legitimate SYN requests.

2.1.3. Reflective attacks

Reflective attacks entail the spoofing of a victim's IP address. The attacker sends requests to a server, which unwittingly directs its response to the spoofed IP address. Fig. 1 depicts an example of a Reflective DDoS attack using IoT devices. Other types of DDoS attacks include TCPreset, UDPStorm, and Topology discovery poison attack [10].

Reflective attacks, by nature, do not directly target IoT devices. Instead, they exploit publicly accessible servers to amplify and redirect traffic to the victim. In a typical home network scenario, where IoT devices have private IP addresses shielded by NAT, the reflective attack would proceed as follows:

1. Initial Compromise and Command and Control (C&C): The attacker first compromises one or more IoT devices within the home network. This can be achieved through various means such as exploiting default credentials, unpatched vulnerabilities, or phishing attacks. Once compromised, these devices can communicate with an external Command and Control server, even in a NAT environment, as outbound connections from private IP addresses are allowed and managed by the router.
2. Exploitation of Public Servers: The attacker utilizes the compromised IoT devices to send requests to public servers (e.g., DNS, NTP) with spoofed source IP addresses. These requests are crafted to elicit large responses. The source IP addresses used in the spoofing are those of the intended victim, which could be an external entity or another device within the same home network.
3. Amplification and Reflection: Public servers, upon receiving these requests, respond with significantly larger payloads directed towards the victim's IP address. This amplification of traffic, combined with the reflection (redirecting the response to an address different from the source), results in a substantial amount of unsolicited traffic overwhelming the victim.
4. Impact on Smart Home Networks: In home networks, where IoT devices are typically behind NAT, the router plays a crucial role. The unsolicited traffic from the public servers is directed towards the home router's public IP address. The router, depending on its configuration and the nature of the attack, may struggle to manage this influx, leading to network congestion or denial of service.

It is important to note that while NAT provides a layer of protection by masking internal IP addresses, it does not inherently prevent compromised internal devices from participating in reflective attacks, nor does it fully shield the home network from becoming the target of such attacks.

2.2. Key considerations for DDoS detection and mitigation solutions

In the context of DDoS detection and mitigation, understanding essential factors is crucial for devising effective solutions. The following items, defined in technical documents such as the Internet Denial of Service Considerations (RFC 4732) [11] and the US-CERT DDoS Quick Guide [12], are considered key factors:

1. Ability to detect and mitigate DDoS attacks from external sources and within the local network (internal attacks) — It is vital to detect attacks from spoofed IP addresses generated by malicious and compromised hosts within the IoT environment, as well as attacks generated from external domains. This will also prevent compromised IoT devices from being used for attacks against external domains.
2. Detection and Mitigation of multiple attacks — The proposed solution should detect multiple types of DDoS attacks, such as Volumetric and Network exhaustion attacks.
3. Protecting and Supporting the SDN control plane from DDoS attacks — The SDN control plane becomes a single point of failure for network devices when overloaded, or when the link between the OpenFlow switch and the SDN controller is exhausted due to an attack. The proposed system should also prevent overloading the switch's TCAM memory by employing effective mitigation techniques.
4. Online Validation — The proposed solution should detect DDoS attacks in real-time without interrupting network operation. Additionally, it should have the ability to validate the solution using various experiments, network topologies, and datasets.

2.3. Security and privacy considerations

We integrate specific measures within our proposed SDN-based DDoS detection and mitigation framework to ensure the privacy of users is maintained. Privacy, a primary concern in home networks, is addressed through the following mechanisms:

1. Data Handling and Processing: All data collection and analysis processes within our framework are designed to comply with privacy-preserving principles. Only essential network traffic metadata, such as packet headers, are used for analysis, avoiding the inspection of payload data that could contain personal or sensitive information. We employ strict data minimization techniques to ensure that only the data necessary for detecting and mitigating DDoS attacks are processed, thereby reducing the exposure of private information.
2. Local Processing and Decision Making: Key components of our framework, particularly the ML models for intrusion detection, operate predominantly within the smart home network. By processing data locally, we avoid external data transmission, thereby reducing the risk of privacy breaches. Decisions regarding network security are made internally within the SDN controller, which resides within the Home's network, further ensuring that sensitive data does not leave the premises.

3. Proposed framework

This section presents our proposed detection framework designed to protect the SDN controller and IoT devices from DDoS attacks. The framework includes a SNORT IDS and trained ML IDS.

3.1. SNORT IDS framework

SNORT [13] is an open-source network intrusion detection system (IDS), that was chosen for its robustness, flexibility, and well-established reputation in network security. Its signature-based detection approach is highly effective in identifying known attack patterns, making it a reliable choice for protecting the SDN controller against common and well-documented threats. Moreover, SNORT's extensive support for custom rules allows for tailored security measures specific to the SDN environment. The decision to use SNORT is also influenced by its widespread adoption in both academic research and industry, which ensures a rich set of resources and community support for its integration and deployment in our framework.

The SDN controller can be targeted via the OpenFlow port or the IP address used for communication with the OpenFlow switch. Therefore, two SNORT rules are created to detect such attacks aimed at the OpenFlow port and the SDN controller's IP address. However, using SNORT IDS to protect IoT devices is often impractical as it necessitates creating multiple rules tailored to each device's characteristics in the network. Moreover, updating rules for vulnerable IoT devices becomes cumbersome with the emergence of new attack forms. Fig. 2 illustrates the SNORT detection framework employed to safeguard the SDN controller from DDoS attacks. The main components of the proposed SNORT IDS detection framework are discussed below:

1. The smart home network comprises numerous IoT devices linked to an OpenFlow switch, each serving a distinct purpose. For instance, an IP camera transmits streams of captured video, while other sensors gauge humidity, temperature, and pressure. Both benign and malicious network traffic can be generated within the smart home network.
2. SNORT IDS, an open-source IDS, categorizes network traffic based on predefined rules in the signature database.

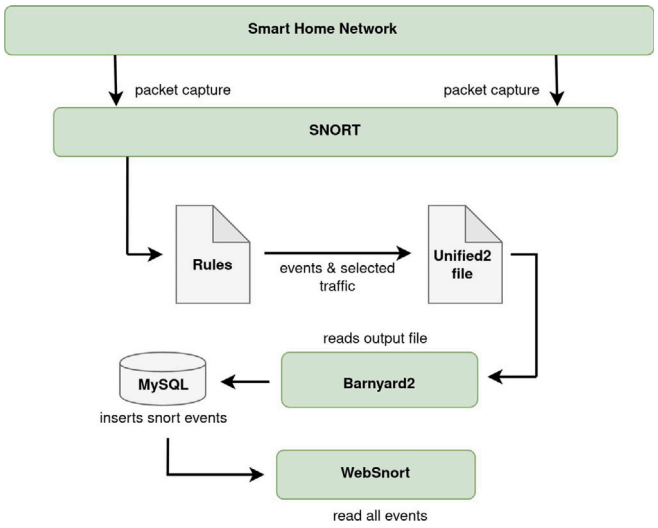

Fig. 2. SNORT IDS detection framework.

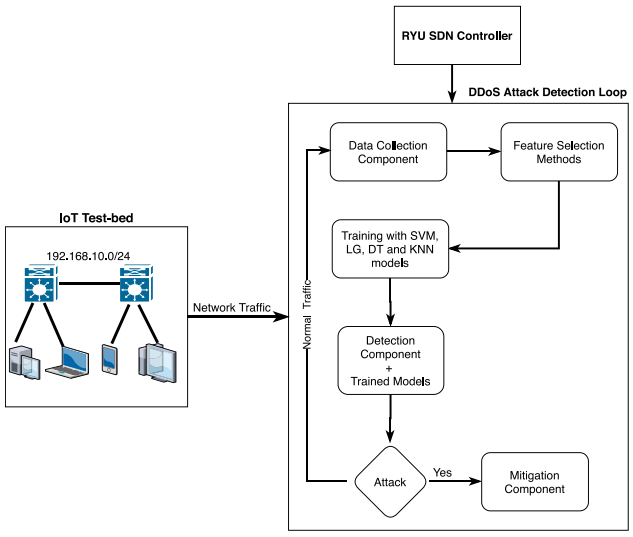

Fig. 3. Components of the system architecture.

3. Rules are written in a single line, featuring the rule header (source and destination IP addresses, protocols, port numbers, and actions) and rule options (specifying the packet portion to inspect and the defined rule action).
4. The Unified2 File contains a packet alert generated from a specific event, including the packet causing the alert and its length in octets.
5. Barnyard2 interprets Snort unified2 binary output files and conducts deep packet inspection.
6. MySQL Database stores interpreted events from Barnyard2, exporting them as CSV or Pcap files for further analysis.
7. WebSnort serves as a web interface for packet analysis, monitoring, and viewing SNORT alerts.

3.2. Detection architecture

The main objective of the detection architecture is the capability to effectively detect DDoS attacks, also, mitigate such attacks in real-time while maintaining continuous network operation of legitimate traffic. The proposed system architecture consists primarily of three components, namely Data Collection, Detection, and Mitigation components, as shown in Fig. 3. All components of the proposed system are applications running on the RYU controller written in Python programming language.

(1) Data Collection: The Data Collection component is tasked with gathering network traffic within the topology. This includes both benign and attack traffic, which is utilized to train ML algorithms. The data is collected using the Flow-based monitoring technique in the RYU Controller. Flow entries in SDN switches, managed by the controller, are used to track network traffic patterns. These entries provide valuable data that are used for detecting potential DDoS attack patterns. When a new flow is detected by the switch, it forwards the flow details to the SDN controller. The controller then analyzes this information as part of the data collection process.

(2) Detection: The DDoS Detection component consists of two elements: ML models, trained and exported from TensorFlow, and a detection agent deployed on the SDN controller. In the subsequent sections of the paper, we delve into the specifics of various ML models that can be integrated as part of the detection component.

(3) Mitigation: The DDoS Mitigation component leverages the OpenFlow protocol's programmability to respond to detected attacks, either by rerouting or dropping traffic from the attacking host. It is worth noting that as the SDN ecosystem progresses with advancements like P4 [14], our proposed mitigation method can seamlessly be integrated into P4 environments. All essential elements required for our method are readily available in P4 environments, offering enhanced flexibility. Through P4, we can further customize and fine-tune packet processing behaviors, including rerouting or dropping traffic, crucial for effective DDoS attack mitigation.

3.3. Real-world deployment considerations

To provide a clearer understanding of the deployment model for our proposed SDN-based DDoS detection and mitigation framework in real-world scenarios, we detail the placement and role of each component within a typical broadband connectivity architecture.

3.3.1. SDN controller deployment

In a real-world deployment, the SDN controller, which is central to our framework, would ideally be situated within the home where the IoT devices are situated. This placement ensures robust connectivity and control over the network infrastructure of the smart home which allows for efficient management of network traffic and security policies.

3.3.2. Data collection and analysis

The data collection component, responsible for gathering network traffic data for analysis, would be deployed at strategic network nodes within the smart home, such as the smart home main router or network switch, providing visibility to the entire network traffic. Collecting data at these points allows for a comprehensive view of the traffic patterns and potential threats, ensuring effective monitoring and analysis.

3.3.3. Machine learning model training and inference

For the training of ML models, we propose utilizing cloud-based services or dedicated servers within the smart home infrastructure. These platforms offer the necessary computational resources for processing large datasets and training complex models. In this paper, we utilized the Google Cloud platform for model training.

Once trained, the inference instances of these models can be deployed on the SDN controller. This integration allows for real-time analysis and decision-making, enabling prompt detection and mitigation of DDoS attacks at control plane of the network.

It is important to note that our system operates exclusively within the smart home infrastructure, ensuring data remains within the premises, thus addressing privacy concerns. Although training may take place on a cloud server or external platform, our method solely extracts network traffic metadata and packet headers, avoiding the use of sensitive or personal packet payload data for training, thereby minimizing privacy risks.

4. Selecting suitable classifier

In the development of our proposed DDoS detection framework, we utilize four distinct supervised ML algorithms, selected for their individual strengths in addressing the challenges of IoT network traffic analysis. Here, we elaborate on the reasoning behind the selection of each model and its effectiveness in our study.

1. Decision Trees: Utilized for their simplicity and efficiency, Decision Trees are binary tree structures that make predictions using both categorical and numeric features, without the need for normalization. This characteristic is particularly advantageous in handling diverse IoT data types. They are renowned for their rapid training and prediction capabilities, making them suitable for real-time attack detection in IoT environments.
2. Logistic Regression: This model is popular for its scalability to a large number of features and efficient distributed training. It functions by classifying numerical feature vectors, making it adept at predicting the probability of network traffic being normal or an attack. In our study, Logistic Regression demonstrated significant efficacy in distinguishing between benign and malicious traffic.
3. Support Vector Machines (SVM): SVM is renowned for its exceptional performance in high-dimensional spaces, a common trait of network traffic data. While training with extensive datasets can be time-consuming, SVM efficiently scores new data points. Our performance evaluation in the later part of the paper shows its accuracy in classifying network traffic within our IoT network context.
4. K-Nearest Neighbors: Known for its straightforward approach, KNN rapidly trains by storing all feature vectors and their labels. During prediction, it identifies the most common label among a test sample's K nearest neighbors. This method proved to be effective in our framework, particularly due to its simplicity and the relatively smaller size of the IoT datasets we analyzed.

It is worth mentioning that our detection method involves a lightweight trained ML model that performs rapid inferences for features extracted from network flows, making it highly scalable. However, the training phase for constructing an effective detection model is a resource-intensive and time-consuming process, requiring powerful servers and extensive data collection. In practical scenarios, such ML models can be trained once using large datasets gathered from real smart home environments and reused multiple times by deployment across several smart home systems.

Given the limited size and quality of the dataset available in our prototype smart home testbed, we made the deliberate choice to exclude deep learning models from our approach. Deep learning models typically excel in tasks requiring extensive data and intricate relationships, elements not readily available in our case study. Therefore, we leave this avenue as a potential area for future research.

Table 1
Feature description before PCA.

Feature	Description	Type
ip.length	Total Length	Numeric
frame.length	Frame Length	Numeric
ip.protocols	Protocol	String
ip.hdr_len	Header Length	Numeric
ip.flags.df	Fragment	Numeric
ip.flags.mf	More Fragments	Numeric
tcp.flags.res	Reserved	Numeric
tcp.flags.ns	Nonce	Numeric
tcp.flags.cwr	Congestion Window Reduced	Numeric
tcp.flags.ecn	ECN-Echo	Numeric
tcp.flags.ack	Acknowledgment	Numeric
tcp.flags.push	Push	Numeric
tcp.flags.reset	Reset	Numeric
tcp.flags.syn	SYN	Numeric
tcp.flags.fin	Fin	Numeric
tcp.windows_size	Calculated window size	Numeric
tcp.srcport	Source Port	Numeric
tcp.dstport	Destination Port	Numeric
tcp.length	Segment Length	Numeric
tcp.hdr.len	Header Length	Numeric
ip.ttl	Time to live	Numeric
tcp.ack	Ack number	Numeric
ip.frag.offset	Fragment Offset	Numeric
tcp.time.delta	Time since first frame	Time offset
class	Traffic Category	Binary

4.1. Feature Selection (FS)

The network traffic captured by the Data Collection component of the proposed detection architecture in our smart home testbed undergoes Feature Selection (FS) analysis before training with the selected ML algorithms discussed in the previous section. The datasets captured from the IoT testbed comprise a total of 29 features, with 2,990,062 packets for attack traffic and 2,668,936 packets for normal traffic. Various techniques were employed to analyze the traffic features, including those detailed below. Tables 1 and 2 present the features before and after Feature Selection, respectively.

4.1.1. Principal Component Analysis (PCA)

This is a feature reduction technique where the dataset is decomposed into principal components. The objective is to transform a large dataset into a smaller one while preserving as much of the original dataset's information as possible. PCA facilitates the visualization and analysis of high-dimensional datasets in a lower-dimensional space, thereby reducing the overall complexity of the detection algorithm.

To ensure that no single feature disproportionately influences the final components, we employed the Standard Scaler class from TensorFlow to center the data by removing the mean and scale it. This normalization procedure ensures that all features contribute equally to the analysis.

4.1.2. Feature selectors class

This method employs three techniques to eliminate redundant selected features from the datasets. The feature selector class comprises the following methods:

1. Identify columns with missing fractions
2. Find feature with only one single unique value
3. Locate collinear features using a correlation coefficient

Missing and Single Unique Values: Zero value features were identified in the captured dataset with missing values above 70%, 80%, and 90% thresholds upon running the feature selector class for missing values. Furthermore, the second method employed is straightforward and involves finding any feature that has only a single unique value. This technique did not identify any features with a single unique value.

Table 2
Feature description after PCA.

Feature	Description	Type
frame.length	Frame Length	Numeric
ip.protocols	Protocol	String
ip.hdr_len	Header Length	Numeric
ip.flags.mf	More Fragments	Numeric
tcp.flags.res	Reserved	Numeric
tcp.flags.cwr	Congestion Window Reduced	Numeric
tcp.flags.ecn	ECN-Echo	Numeric
tcp.flags.push	Push	Numeric
tcp.flags.reset	Reset	Numeric
tcp.flags.syn	SYN	Numeric
tcp.flags.fin	Fin	Numeric
tcp.windows_size	Calculated window size	Numeric
tcp.srcport	Source Port	Numeric
tcp.dstport	Destination Port	Numeric
ip.ttl	Time to live	Numeric
tcp.ack	Ack number	Numeric
ip.frag.offset	Fragment Offset	Numeric
tcp.time.delta	Time since first frame	Time offset
class	Traffic Category	Binary

The results indicate that each column contains relevant information using different threshold values.

Pearson Correlation: This method finds pairs of features that are linearly related. The Pearson correlation ranges between −1 and 1, with a value of 0 indicating no correlation between the features. A value closer to 0 indicates a weak correlation. A value of exactly 1 indicates a strong positive correlation and a value closer to −1 implies a strong negative correlation.

In the experiments, a correlation threshold of 90% was selected. Utilizing this threshold for each pair of features, the Pearson correlation identifies features to be removed from the dataset. Typically, the features identified for removal are those that come last in the column order.

4.2. Network topology

The smart home IoT testbed's network topology, illustrated in Fig. 4 features a RYU SDN controller linked to a TP-Link switch that serves as a DHCP server for the controller and the OpenFlow switches connected to the same TP-link switch.

The Ryu controller was selected due to its compatibility with the OpenFlow protocol, ease of use, and active development community. Its open-source nature and the availability of comprehensive documentation make it a suitable choice for academic research, allowing for transparent and reproducible results. Additionally, Ryu's lightweight design and straightforward programming interface facilitate the integration of custom applications like our DDoS detection and mitigation modules. This flexibility, combined with its performance and scalability, makes the Ryu controller an optimal choice for our research purposes.

The RYU controller is installed on a server running Ubuntu 18.04.1 LTS, Intel i7 processor with 8 GB of RAM. Two Raspberry Pis are configured to serve as OpenFlow switches, with IoT devices subsequently connected to these switches. Among these devices is an IP camera equipped with a motion sensor, which captures movements and uploads image streams using the FTP protocol to an FTP server hosted on a network-connected device. Additionally, temperature, humidity, and pressure sensors gather data from a living room and transmit it via the MQTT protocol to a dashboard for visualization, utilizing Freeboard [15].

Google Chromecast is connected to a SONY Smart TV, while a Google Home Mini is linked to the same network. Table 3 provides a list of devices within the smart home network. A host connected to the OpenFlow switch executes multiple TCP SYN DDoS attacks against the SDN controller on port 6633, the IoT sensors, and conducts another attack on the IP Camera by uploading streams of captured images. These DDoS attacks are conducted utilizing the *Xerxes* [16] and *Hping3* [17] tools.

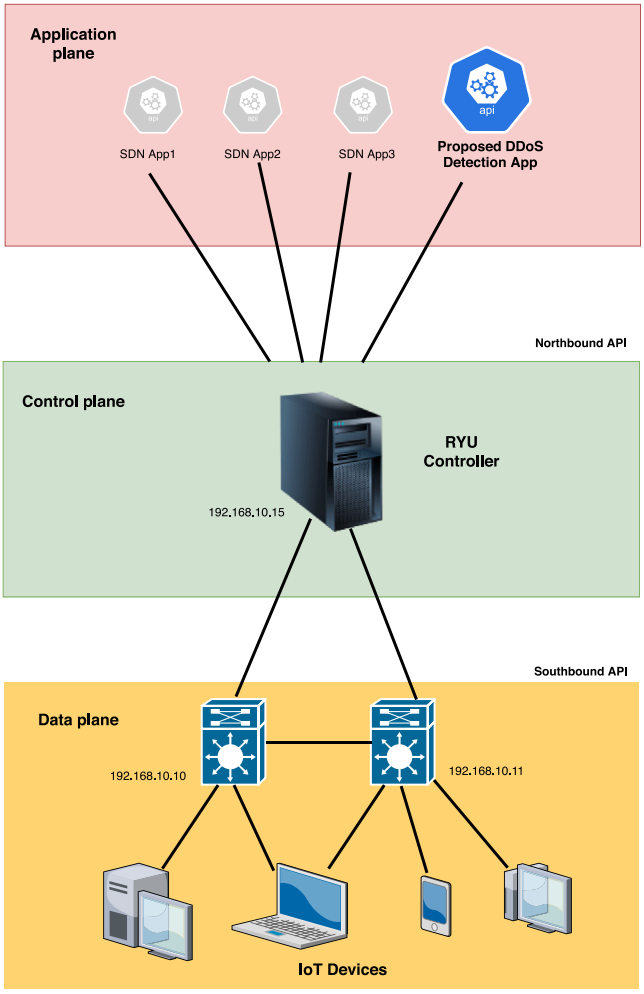

Fig. 4. SDN-IoT topology.

Table 3
List of IoT devices.

IoT device	Connection type
Google Home Mini	Wireless
Smart TV	Wired
IP Camera	Wired
Sensors (Temp. Humidity, Pressure)	MQTT
Smart Plug	Wireless
Google Chromecast	Wireless

TensorFlow is an open-source library developed by Google for numerical computation and large-scale machine learning. The captured network traffic is trained using TensorFlow on a Tesla K80 GPU, which features 2496 CUDA cores and 12 GB GDDR5 VRAM. The trained models are exported from TensorFlow and utilized to assess the performance of the proposed detection framework. Four supervised machine learning algorithms were chosen to classify DDoS attack traffic from regular traffic in the smart home network. These classifiers were trained using a training set comprising 70% of the dataset, including both benign and attack traffic, while the remaining data served as a validation set. The experiment was repeated ten times to ensure the training results derived from the datasets are free from uncertainty.

```
0] [Priority ID: 0] {TCP} 133.86.175.53:3120 -> 10.10.10.10:6633
08/21-05:26:01.639836  [**] [3:10000003:1] SDN Controller Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 133.86.175.53:3120 -> 10.10.10.10:6633
08/21-05:26:01.639908  [**] [4:10000004:1] TCP SYN Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 247.116.62.175:3121 -> 10.10.10.10:6633
08/21-05:26:01.639908  [**] [3:10000003:1] SDN Controller Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 247.116.62.175:3121 -> 10.10.10.10:6633
08/21-05:26:01.639985  [**] [4:10000004:1] TCP SYN Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 134.146.153.211:3122 -> 10.10.10.10:6633
08/21-05:26:01.639985  [**] [3:10000003:1] SDN Controller Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 134.146.153.211:3122 -> 10.10.10.10:6633
08/21-05:26:01.640053  [**] [4:10000004:1] TCP SYN Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 183.88.253.172:3123 -> 10.10.10.10:6633
08/21-05:26:01.640053  [**] [3:10000003:1] SDN Controller Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 183.88.253.172:3123 -> 10.10.10.10:6633
08/21-05:26:01.640121  [**] [4:10000004:1] TCP SYN Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 26.108.163.153:3124 -> 10.10.10.10:6633
08/21-05:26:01.640121  [**] [3:10000003:1] SDN Controller Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 26.108.163.153:3124 -> 10.10.10.10:6633
08/21-05:26:01.641867  [**] [4:10000004:1] TCP SYN Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 107.31.27.124:3125 -> 10.10.10.10:6633
08/21-05:26:01.641867  [**] [3:10000003:1] SDN Controller Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 107.31.27.124:3125 -> 10.10.10.10:6633
08/21-05:26:01.641948  [**] [4:10000004:1] TCP SYN Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 196.249.32.22:3126 -> 10.10.10.10:6633
08/21-05:26:01.641948  [**] [3:10000003:1] SDN Controller Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 196.249.32.22:3126 -> 10.10.10.10:6633
08/21-05:26:01.642017  [**] [4:10000004:1] TCP SYN Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 162.230.135.4:3127 -> 10.10.10.10:6633
08/21-05:26:01.642017  [**] [3:10000003:1] SDN Controller Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 162.230.135.4:3127 -> 10.10.10.10:6633
08/21-05:26:01.642091  [**] [4:10000004:1] TCP SYN Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 52.131.230.174:3128 -> 10.10.10.10:6633
08/21-05:26:01.642091  [**] [3:10000003:1] SDN Controller Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 52.131.230.174:3128 -> 10.10.10.10:6633
08/21-05:26:01.642181  [**] [4:10000004:1] TCP SYN Attack [**] [Classification ID: 0] [Priority ID: 0] {TCP} 174.182.214.204:3129 -> 10.10.10.10:6633
```

Fig. 5. TCP SYN and SDN controller attacks.

4.3. Attack scenarios

As mentioned earlier, two different attack tools, *Hping3* [17] and *Xerxes* [16], are employed to execute the attacks. Each tool adopts distinct approaches to conduct DDoS attacks, offering various attack modes and supporting different attack types. These tools are utilized to execute actual DDoS attacks against a specified target. To prevent overfitting with the machine learning algorithms, we utilized both tools in different attack modes against the targeted IoT devices. Subsequently, the network traffic was captured for analysis using the selected machine learning algorithms.

The *Xerxes* tool offers an automated method for executing DDoS attacks, allowing for the launch of multiple attacks against the same target or multiple targets simultaneously. On the other hand, the *Hping3* tool can send large volumes of TCP SYN attack traffic while spoofing the source IP addresses, ensuring that the attack appears random to the target and originates from multiple sources [18].

We conducted DDoS attacks in multiple scenarios during the experiment. In Scenario I, the SDN controller was targeted without configuring the SNORT IDS in the IoT testbed. Multiple hosts were utilized to launch TCP SYN attacks directed at the controller's IP address and port number. Additionally, the attack tool employed generated TCP SYN attacks from spoofed random IP addresses. The outcome of this attack was a complete loss of network functionality, causing the entire IoT testbed to go offline as the SDN controller served as a single point of failure. Packet loss in this scenario was observed to be 100%. This scenario was designed to illustrate the devastating impact of the attack on the network controller in the absence of any security mechanisms.

To avert the scenario represented in Scenario I, we carried out another attack on the SDN controller similar to that in Scenario I. However, in Scenario II, the SNORT IDS was configured to safeguard the controller from such attacks. Two rules were established in the SNORT database to monitor the controller's IP address and port number. Any traffic matching these rules initiated from an external network would trigger the SNORT IDS to identify it as an attack, generating an alert on the SNORT console, as illustrated in Fig. 5.

In Scenario III, hosts within the smart home network are employed to launch attacks against the IP camera and sensors responsible for streaming images and collecting environmental conditions data from the home. An attack script automates the attacks during regular network operations; this script dispatches multiple traffic streams to the designated targets using randomly spoofed IP addresses. Referring to the switch flow table depicted in Fig. 6, an output action is appended to each flow to mirror all network packets to port 5. The network traffic in this scenario is captured during regular network operation and post-attack. It is then converted to a format suitable for using the Data collection component described in Section 3.2. Finally, the captured network traffic is utilized to train the selected ML algorithms.

5. Performance evaluation

In this section, we evaluate the performance of our proposed method, along with the evaluation of the selected ML models, using both the smart home IoT testbed dataset and other existing datasets. First, we analyze the performance of ML models utilizing the IoT testbed. Then, we compare their performance with other publicly available datasets. Finally, we report the results of evaluating the signature-based IDS performance to protect the SDN controller.

We utilize Google TensorFlow [19] platform to train ML models. For the smart home IoT testbed, the captured network traffic is imported from the data captured in the smart home IoT testbed. Additionally, we evaluate our models using standard IoT DDoS attack datasets published by the Canadian Institute for Cybersecurity (CIC) [20] and the Cyber Range Lab of the Australian Center for Cyber Security (ACSS) [21].

5.1. Datasets

Before we delve into the evaluation of ML methods, we provide the details of employed datasets. We employed our own captured IoT testbed dataset derived from our smart home IoT setup alongside two DDoS datasets furnished by the Canadian Institute of Cybersecurity [20] and the Cyber Range Lab of the Australian Center for Cyber Security (ACSS) [21]. Both of these public datasets have been extensively employed in security research [22]. Below, we present detailed information about the datasets.

IoT testbed Dataset: This dataset is collected from the OpenFlow switches of the testbed, ensuring the capture of the total network traffic within the IoT testbed. This process is conducted using the data collection component of the proposed detection framework discussed in Section 3.2. The captured network traffic is stored in PCAP format over a 24-hour period while the network operates under regular conditions and when scripts are employed to launch TCP SYN DDoS attacks generated from randomly spoofed IP addresses, as explained earlier. Subsequently, the PCAP file is converted to a CSV file format using the same data collection component. This CSV file is utilized to train the ML models in TensorFlow. The IoT testbed captured traffic comprises a total of 29 features, with attack traffic totaling 2,990,062 packets and regular traffic 2,668,936 packets. However, after conducting traffic feature extraction, the number of features is reduced to 19, as shown in Table 2.

UNSW-NB15 Dataset: This dataset was created in the Cyber Range Lab of the Australian Center for Cyber Security (ACCS), capturing approximately 100 GB of raw network traffic containing nine different attack types: Fuzzers, Reconnaissance, Backdoors, DDoS, Exploits, Shellcode, and Worms. A hybrid dataset was constructed, comprising both real modern activities and synthetic contemporary network attacks. The dataset comprises about two million records stored in four CSV files. The network topology and a complete description of the dataset are published in [21]. This dataset has been utilized in

SW_963354558773

Flow Table 0

PRIORITY	MATCH FIELDS	COOKIE	DURATION	IDLE TIMEOUT	HARD TIMEOUT	INSTRUCTIONS	PACKET COUNT	BYTE COUNT	FLAGS
1	in_port = LOCAL eth_dst = 28:d2:44:79:a9:4e	0	4650	0	0	OUTPUT:1 OUTPUT:5	54	3556	0
1	in_port = 5 eth_dst = 28:d2:44:79:a9:4e	0	4650	0	0	OUTPUT:1 OUTPUT:5	8	336	0
1	in_port = 1 eth_dst = 00:e0:4c:68:01:35	0	4488	0	0	OUTPUT:LOCAL OUTPUT:5	2091985	163440024	0
1	in_port = LOCAL eth_dst = b8:27:eb:5e:e9:86	0	4297	0	0	OUTPUT:3 OUTPUT:5	156	14784	0
1	in_port = 5 eth_dst = b8:27:eb:5e:e9:86	0	4297	0	0	OUTPUT:3 OUTPUT:5	12	504	0
1	in_port = 3 eth_dst = 00:e0:4c:68:01:35	0	4297	0	0	OUTPUT:LOCAL OUTPUT:5	149	14488	0
0	ANY	0	4701	0	0	OUTPUT:CONTROLLER	623	252334	0

Fig. 6. Switch flow table.

numerous studies in the literature to evaluate the performance of various proposed algorithms.

CICDDoS2019 Dataset: The CICDDoS2019 dataset contains various recent types of DDoS attacks, encompassing more than 1 million benign traffic instances and over 30 million instances of attack traffic. The dataset includes 26 different types of network attacks such as NTP, DNS, LDAP, TCP SYN, and NetBIOS. Originally provided in PCAP format, the authors utilized the CICFlowMeter to extract features from the PCAP files and convert them into CSV format. The dataset was captured over a 24-hour period, resulting in a storage size of 4.6 GB. Notably, the dataset utilized for the training phase is entirely distinct from that used for the testing phase. For comprehensive details, including network topology and dataset analysis, refer to [20].

5.2. Performance of machine learning models utilizing IoT testbed dataset

The first algorithm we analyze is the Decision Trees model, which can solve regression and classification problems. Selecting the attribute to place as the root of the tree is considered a complicated step. Selecting a random node will result in low classification accuracy. To solve the selection attribute, we used the Gini index criteria. The Gini index is a cost function used to evaluate splits in datasets, calculated by subtracting the sum of each class's squared probabilities from one, as denoted in Eq. (1), where P_i is the probability of network traffic classified to a particular class.

$$Gini = 1 - \sum_{i=1}^{c}(P_i)^2 \tag{1}$$

The Gini index calculations are in two steps: first, the sub-nodes are calculated using the equation above for regular traffic and attack traffic. Then, the split's Gini index is also calculated using the weighted score of each node's particular split. The Gini index degree ranges between 1 and 0, with 1 denoting randomly distributed traffic across the two classes (attack and benign) and 0 representing all the network traffic belonging to a particular class. A well-balanced distribution of the network traffic into the two classes has a Gini index of 0.5. Another common problem with Decision trees is overfitting, which can result in a 100% accuracy on the training data, eventually affecting the classification of any unseen data. We counter this issue by using pruning, which continuously removes the decision nodes until the overall accuracy is not affected.

The Decision Tree algorithm performs exceptionally well in Precision, Recall, and F1-Score, with a detection accuracy of 98.93%. The algorithm achieves excellent results using the full feature set from the dataset, with a slight increase in detection accuracy after conducting the feature selection to 99.57%. These results can be attributed to the algorithm's use of Information Gain in selecting the best feature that splits the data during the construction of the tree.

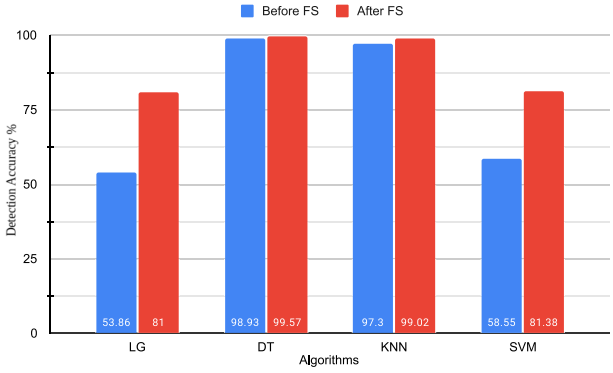

Fig. 7. Classification of detection accuracy.

The second algorithm analyzed is K-Nearest Neighbors (KNN), which is versatile in solving both classification and regression predictive problems. The KNN algorithm depends on the assumption that similar data points are always in proximity to each other [23]. We used the Minkowski distance function given in Eq. (2) to calculate the K values, where x and y denote the distance between two points, and q is an integer between x and y.

$$Minkowski = \left(\sum_{i=1}^{k}\left(\left|x_i - y_i\right|\right)^q\right)^{1/q} \tag{2}$$

To select a K value for the dataset, the algorithm is run several times with different values. Finally, the value that reduces the percentage of error while accurately making predictions with unseen network traffic is chosen as the K value.

K-nearest neighbor closely follows Decision Trees as the second-best performer in all metrics, achieving a detection accuracy of 97.30% before feature selection and increasing to 99.02% after reducing the number of features. This notable improvement far surpasses the performance of Logistic Regression and Support Vector Machine models.

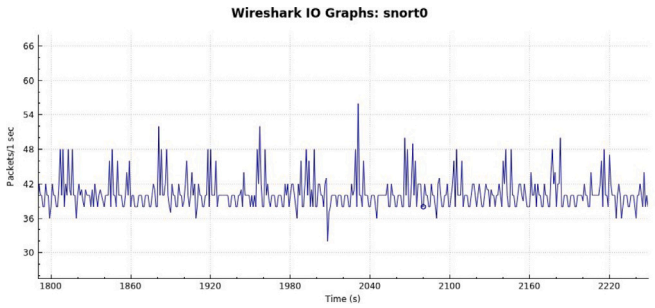

Fig. 8. Benign network traffic.

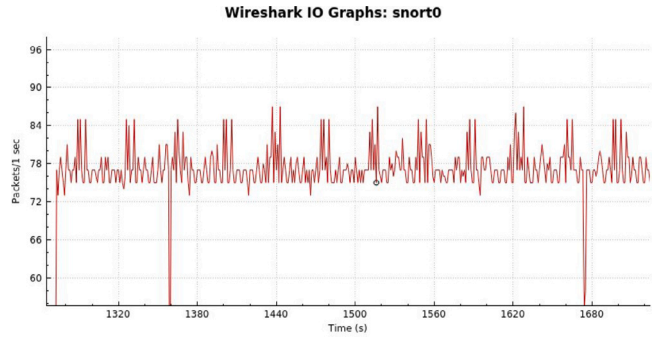

Fig. 9. Attack network traffic.

The next algorithm we use for DDoS attack detection within our proposed framework is Logistic Regression. It supports categorical dependent variables and uses the sigmoid function to handle outliers, as shown in Eq. (3), where p is the probability estimate between 0 and 1, x is the algorithm's prediction, and e is the base of the natural logarithm. The sigmoid function ensures that the final prediction value is between 0 and 1, unlike in Linear Regression, which can produce values beyond 1 or less than 0. Additionally, the sigmoid function aids in making the final classification between an attack and regular traffic. However, the inclusion of independent variables can increase the amount of variance in the logistic regression model, leading to overfitting.

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{3}$$

Logistic Regression performs poorly with an initial detection accuracy of 53.86% with 29 features. However, after carrying out the principal component analysis (dimension reduction technique) of the features, the detection accuracy improves significantly to 81%.

The final algorithm used to identify new attacks from the smart home network after training the captured network traffic model is the Support Vector Machine (SVM). It requires low computational power and is also employed for regression and classification tasks. To distinguish between an attack and benign traffic within the data points, the algorithm creates a hyperplane (decision boundaries) between the two classes [23]. The primary objective is to identify a hyperplane that maximizes the margin; support vectors (data points closer to the hyperplane) are utilized to achieve this goal. In this work, we employ the Hinge loss function to maximize the margin between the hyperplane and the data points. Additionally, the regularization parameter is used to balance the maximized margin and the loss, as shown in Eq. (4).

$$loss function = \beta + C \sum_{i=1}^{N} \xi_i \tag{4}$$

Support Vector Machine, before performing principal component analysis, performs rather poorly with an accuracy of 58.55%; however, after dimensional reduction of the features, the accuracy is improved to a detection rate of 81.38%. The detailed comparison of results is shown in Fig. 7.

5.3. Datasets comparison results

The comprehensive evaluation comparison involving the CICDDoS2019, UNSW-NB15, and IoT testbed datasets is presented in Table 4. Employing identical model parameters and feature selection techniques as those used on the IoT testbed dataset, we extend our analysis to the CICDDoS2019 and UNSW-NB15 datasets to facilitate comparison across larger-scale environments.

Beginning with the UNSW-NB15 dataset, the results show similar performance trends to those observed with other datasets. Notably, the Decision Tree algorithm emerges as the top performer, achieving a detection accuracy of 98.2%, with the K-Nearest Neighbor being the second best performer with 93.78%. The SVM and Logistic Regression algorithms exhibit lower performance, with detection accuracies of 89.19% and 89.36%, respectively.

The next dataset we evaluated with the detection model is the CICDDoS2019 dataset. This dataset is used in almost every literature for performance evaluation of various proposed detection algorithms. As with the other datasets, the Decision Tree algorithm also performs exceptionally well with this dataset. In terms of detection accuracy, the algorithm achieves a 99.95% accuracy with the CICDDoS2019 dataset compared to 99.57% for the captured dataset and 98.20% for the UNSW-NB15 dataset. The next well-performing algorithm is the K-Nearest Neighbor with 99.94% detection accuracy. With the UNSW-NB15 dataset, the detection accuracy is 93.78%, while with our captured dataset, it reaches 99.02%. Note that the training and testing stages took about one hour and forty minutes with the CICDDoS2019 dataset, while it took only five minutes with the UNSW-NB15 and approximately two minutes with the dataset from our IoT testbed. SVM and Logistic Regression again underperformed with the CICDDoS2019 dataset, as observed with the UNSW-NB15 and the captured IoT dataset. The detection accuracy for both SVM and Logistic Regression using the CICDDoS2019 dataset is 95.7% and 98.4%, respectively.

Considering the three datasets, based on the weighted average of the evaluation metrics, it is observed that the Decision Tree algorithms have the highest detection accuracy, Precision, Recall, and F1-Score as shown in Table 4. It can be concluded that this is the most suitable algorithm for this problem, while the Support Vector Machine model and Logistic Regression are the least suitable for DDoS attack detection using our proposed approach. These results are also in agreement with the analysis of the CICDDoS2019 Dataset conducted by [20], with the ID3 algorithm — a variant of Decision Tree algorithm outperforming the other algorithms used in the analysis such as Naive Bayes and Logistic Regression, which is the worst performing algorithm in their analysis.

5.4. Signature-based IDS performance

Our performance evaluation shows that DDoS attacks towards the SDN controller are timely detected and subsequently mitigated using the SNORT IDS and SDN Controller. The detection is based on the defined rules in the SNORT IDS database. Furthermore, alerts of detected attacks are captured and logged in real-time into a separate database for further analysis. The SNORT console shows the alerts of respective incoming attacks on the SDN Controller IP address and port number, as shown in Fig. 5. Once the SNORT IDS detects an attack, the RYU controller instructs the OpenFlow switch to create a new flow rule to learn the MAC address of the attacker(s) to avoid a subsequent Flood attack from the learned MAC address. Then, the RYU controller installs another flow rule in the switch to drop the packet entirely. To prevent a Content Addressable Memory (CAM) table attack — which is an attack where the switch learns and saves thousands of spoofed random MAC addresses generated by random sources in its memory,

Table 4
Evaluation with standard dataset.

Dataset	Algorithms	Detection accuracy	Precision	Recall	F1-Score
UNSW-NB15	Logistic Regression	89.36	90	89	89
	Decision Tree	**98.20**	**98**	**98**	**98**
	K-Nearest Neighbor	93.78	94	94	94
	Support Vector Machine	89.19	90	89	89
CICDDoS2019	Logistic Regression	98.4	**100**	**100**	**100**
	Decision Tree	**99.95**	**100**	**100**	**100**
	K-Nearest Neighbor	99.94	**100**	**100**	**100**
	Support Vector Machine	95.7	**100**	**100**	**100**
Captured Dataset	Logistic Regression	81	83	81	81
	Decision Tree	**99.57**	**100**	**100**	**100**
	K-Nearest Neighbor	99.02	99	99	99
	Support Vector Machine	81.38	84	81	81

the switch or security endpoint employs a whitelist of legitimate MAC addresses that are saved in the switch's memory. With multiple or groups of devices attacking the SDN controller, the rules defined in the SNORT IDS database will detect such attacks. A packet sniffer tool called Wireshark[1] is used on SNORT IDS port 5 to capture real-time traffic transfer rate during regular and attack network operation, as shown in Figs. 8 and 9. During regular network traffic transfer, the average packets sent per second between the IoT devices were well below 6,000 packets per second. However, after conducting the attack, the average packet rate received increased significantly, close to 90,000 packets per second, as the SDN controller became flooded with TCP SYN attack traffic. Also, communication between the IoT devices during the attack was severely affected, with an average round-trip time between the IP Camera and the FTP server at 19.64 ms, resulting in 39% packet loss, while during regular network operation, the average round-trip time for the same devices is at 0.78 ms. During the experiments, increasing the rate of the DDoS attacks without the SNORT IDS configured and detection rules enabled, the attack quickly exhausts the SDN controller's processing capability and effectively takes the controller offline, resulting in a 100% packet loss.

6. Discussions and future directions

In light of the evolving landscape of IoT security, the implications of using AI and ML become increasingly significant. Our research has demonstrated the instrumental role of ML algorithms in enhancing the detection and mitigation of DDoS attacks in SDN-enabled smart home networks. However, the rapid advancement of IoT technologies and the diversification of IoT devices introduce both challenges and opportunities for AI and ML applications.

When developing AI or ML systems or algorithms, it is imperative to address potential security risks and implement appropriate safeguards against threats or vulnerabilities. While in line with much of the literature, we utilized ML algorithms for DDoS attack detection in this work, we did not delve into the potential security risks associated with employing ML techniques. Further research is required to evaluate any risks inherent in utilizing ML methods to perform DDoS attack detection and mitigation.

Our next phase involves broadening the scope of IoT devices included in the experiments to expand the research's scale and assess the dataset with deep learning algorithms. While we employed supervised learning models in this study, necessitating a labeled dataset – an exhaustive and complex endeavor – we recognize the importance of exploring alternative clustering and unsupervised algorithms that do not rely on labeled data.

Moving forward, our focus will extend to feature selection and analysis, delving into the dataset's key features that significantly influence the clustering of benign network traffic and attack traffic within a smart home network. Future work in this area should prioritize the adaptability and scalability of ML models to accommodate a broader range of IoT devices and scenarios. This includes exploring real-time learning and adaptive algorithms capable of responding to new, unforeseen attack patterns and behaviors, thereby fortifying the resilience of smart home networks against an ever-evolving threat landscape.

NIST's recent lightweight cryptographic standards [24] signify a major step forward in cybersecurity, especially for resource-constrained IoT devices. These standards, tailored for devices with limited resources, are vital for ensuring strong security in IoT setups. The endorsed lightweight cryptographic algorithms by NIST aim to provide robust security without burdening devices, making them ideal for smart home IoT gadgets with limited processing power or battery life.

Besides adopting lightweight cryptographic standards as in [25], addressing fault attacks is crucial for IoT systems. Fault attacks exploit errors to breach security, posing a significant risk to IoT devices. As IoT devices become more widespread, the threat of such attacks increases. Future research should focus on integrating fault detection and mitigation into IoT security to prevent unauthorized access and manipulation of sensitive data.

Another important aspect is considering post-quantum cryptography (PQC) for securing IoT frameworks. PQC offers resilience against quantum computing threats, ensuring long-term security. Future iterations of security systems should assess the application of PQC algorithms, including compatibility with existing ML models and enhancing data transmission security in smart home networks.

Given these considerations, our proposed framework can incorporate both NIST's lightweight cryptographic standards and fault attack mitigation strategies. This integration will significantly enhance the overall security posture of smart home IoT environments. Our future work focuses on evaluating the effectiveness of lightweight cryptographic algorithms in our proposed SDN-based IoT environment and developing fault detection mechanisms as an integral part of the security framework. By doing so, we can ensure that our system not only addresses current security challenges of DDoS attacks but is also prepared for emerging threats in the rapidly evolving landscape of IoT security.

7. Related work

7.1. DDoS detection in IoT networks

Several studies address DDoS detection and mitigation in IoT networks, each with distinct approaches. [26] targets Application Layer DDoS attacks with a time series prediction model, achieving high detection rates and low false positives compared to [27,28]. Unlike their focus on the Application layer, our approach spans the network layer, covering all network endpoints. [29] worked on a lightweight method to detect DDoS attacks based on traffic flow features using the

[1] https://www.wireshark.org/

Self Organizing Maps (SOM) algorithm. The evaluation of the proposed detection system is done using the KDD-99 dataset. However, the KDD-99 dataset has been proven to have several underlining issues such as redundant and duplicate records in the training set, which results in a biased classifier towards the more frequent records [30]. [31] proposed a DoS detection system based on signature-based intrusion detection called Suricata. This method can only detect known attacks in the signature database and requires a continuous update of the database. [32] presents a network of infected IoT devices connected to an SDN-controlled fog network. Rate Limiting and Threshold Random walks with credit-based rate-limiting algorithms are used for DDoS detection and mitigation. In these papers, the proposed algorithms used signature-based or statistical comparison, which performs well in only specific situations.

Similar to our work, [33] also used the SDN controller and used Support Vector Machine, K-nearest neighbors, and Random Forest algorithms to detect DDoS attacks. This work is closely related to this paper; however, we specifically focuses on smart home scenarios and integrate a Signature-based detection technique alongside ML models to protect both the SDN controller and the IoT devices. In the experiment, simulated IoT devices are used as hosts in an SDN Mininet environment, and traffic behavior can be very different from actual IoT devices.

The IoT-IDM framework [34] employs a host-based IDS technique, necessitating setup for each IoT device in a smart home, requiring profiling and registration in a device manager. In contrast, our approach deploys the detection algorithm at the network layer, monitoring IoT device traffic within the smart home network. Another method proposed by [35] positions software-based managers between the IoT network and the gateway router to detect and mitigate DDoS attacks, effective for small-scale IoT deployments like smart homes or buildings.

Similarly, [36] utilizes SDN architecture for dynamic attack detection and mitigation in IoT networks, aiming to prevent attacks at the network level rather than the device level. They propose SoftThings framework, employing a supervised ML algorithm requiring constant retraining and labeled datasets for normal and malicious traffic. Additionally, [37] introduced IoT-specific network DDoS detection using ML algorithms, showing effective detection with accuracy ranging from 0.91 to 0.99, albeit lacking evaluation against standard datasets. Furthermore, [38] presented a method for securing smart homes utilizing SDN and low-cost traffic classification, employing a random forest classifier trained on network traffic data to detect attacks such as DNS-based attacks, port scans, and malware infections. While effective, the approach has limitations including potential false positives and the need for ongoing algorithm retraining.

[39] proposed an SDN-based system for dynamic DoS detection in IoT networks using statistical analysis and ML. They employ OpenFlow protocol and Mininet emulator for evaluation. [40], proposed entropy analysis for DDoS detection in SDN-IoT, but faces challenges like false positives and added computation. [41] also proposed an algorithm that measures entropy implemented in POX [42] SDN controller; the authors claimed their algorithm is lightweight and uses fewer resources, also the algorithm detects and mitigates DDoS attacks promptly, the POX controller terminates processing of the malicious traffic once an attack is detected.

[43] proposed ML for DDoS detection in IoT networks via SDN-cloud setup. It relies on cloud resources for monitoring and may lack real-time suitability due to ML processing delays. [44] proposed an approach to combat DDoS and MitM attacks in SDN-based IoT networks using belief theory and correlation analysis. Though effective, it overlooks other potential threats. [45] propose a framework for real-time detection and mitigation of DDoS attacks in stateful SDN-based IoT networks using ML. Despite its promise, the approach faces limitations in evaluation, scalability, resource requirements, and SDN dependency. Further research is needed to validate its effectiveness under diverse network conditions. In [46], a Snort-based secure edge router is proposed for smart homes, providing protection against network attacks through attack detection and VPN encryption. The paper offers a detailed evaluation, showcasing the solution's efficacy. However, the proposed solution may not be effective against novel or advanced attacks, and it may not be suitable for large or complex smart home environments that require more comprehensive security solutions.

In [47], an intelligent DDoS attack detection model using a tree-based approach and the Gini index feature selection method is presented. Trained on network traffic features, the model achieves high accuracy in detecting DDoS attacks, outperforming other ML techniques. However, limitations include assumptions about dataset balance and lack of real-world testing. Additionally, comprehensive evaluation under various attack scenarios and traffic conditions is lacking. [48] proposes a framework for DDoS attack detection in an SDN-based IoT environment using a hybrid classifier of decision trees and K-nearest neighbors (KNN). Evaluated on a simulated SDN-based IoT setup, the hybrid classifier demonstrates superior accuracy and false positive rates compared to other ML and deep learning techniques.

While previous works focused on enhancing accuracy or response times, they face limitations in smart home environments due to unique configurations and traffic patterns. Our research addresses these limitations by introducing novel strategies and refining existing methodologies. We utilize SVM, Logistic Regression, Decision Trees, and KNN algorithms, selected for their effectiveness in structured datasets like network traffic, particularly suited for diverse smart home interactions. Our approach ensures real-time processing capabilities critical for effective DDoS attack mitigation and demonstrates improved adaptability across various attack types. Through experimental analysis, we provide comparative data showcasing the enhanced performance of our approach, validating its effectiveness in improving smart home network security. While our research represents a significant step forward, we acknowledge limitations and anticipate further refinement and expansion of our findings, aligning with the evolving network security landscape. Table 5 summarizes the proposed solutions and methodologies in the literature.

7.2. Post-quantum and cryptographic methods

The work by [56] delved into the integration of post-quantum authentication mechanisms within the MQTT protocol, a critical component in IoT communications. The authors scrutinize the current vulnerabilities of MQTT in the face of quantum computing advancements and propose solutions to reinforce its security. [57] presented an innovative approach to implementing Curve448 and Ed448 algorithms in the wolfSSL cryptographic library, with a special emphasis on Cortex-M4 processors. [58] proposed the Supersingular Isogeny Key Encapsulation (SIKE) algorithm for FPGA platforms, its relevance in the post-quantum cryptography landscape, and the importance of FPGA in cryptographic applications, particularly for IoT devices. [59] provided a broad overview of the transition to post-quantum security, identifying both opportunities and challenges that arise from this shift.

[60] explored FPGA-based error detection strategies for the WG-29 stream cipher, emphasizing the importance of hardware-level security enhancements. Similarly, [61] investigated error detection mechanisms in the QARMA block cipher, aligning with the imperative for robust cryptographic systems in IoT networks. Additionally, [62] showcased FPGA-based optimization algorithms, vital for resource-efficient computational tasks crucial in IoT security frameworks, including DDoS mitigation. Furthermore, [63] contributed by integrating the Khumbelo function with the Camellia algorithm to reinforce IoT device security against cyber-attacks. Lastly, [64] focused on detecting and mitigating vulnerabilities in the Midori cipher, providing insights into safeguarding cryptographic systems from active side-channel attacks.

Our work is different and can be complementary to these studies as we focus on network traffic analysis instead of cryptographic approaches.

Table 5
Summary of existing literature.

Proposed work	Detection technique	#1	#2	#3	#4
[49]	Cosine Similarity	×	×	×	✓
[50]	Flow Filtering	×	×	✓	×
[39]	SDN-based	✓	×	✓	✓
[43]	Machine Learning	✓	×	✓	×
[51]	Flow Filtering	✓	✓	×	×
[45]	Multi-layer Approach including Machine Learning	✓	×	✓	×
[48]	Machine Learning (Hybrid classifier)	✓	×	✓	✓
[36]	SDN-Fog	✓	×	×	×
[37]	Machine Learning	×	✓	×	×
[38]	Traffic Classification based on Support Vector Machine	✓	✓	×	✓
[52]	Machine Learning	✓	✓	×	×
[41]	Entropy measurement	✓	✓	×	×
[47]	Decision tree with Gini index feature selection	✓	×	✓	✓
[53]	Flow-base SNORT IDS	×	×	×	✓
[46]	SNORT	✓	✓	×	✓
[54]	Machine Learning	✓	✓	✓	×
[44]	Belief-Based Secure Correlation	✓	×	✓	✓
[55]	Count-based Rate Limiting	×	×	✓	✓
Proposed work	SDN-Machine Learning-SNORT	✓	✓	✓	✓

1. Detect internal and external source attacks. 2. Detect multiple types of attacks. 3. Protect the IDS from attacks itself. 4. Real-time DDoS detection.

8. Conclusions

In this paper, we presented a framework for real-time detection and mitigation of DDoS attacks in SDN-enabled smart home networks, utilizing traditional ML models such as SVM, Logistic Regression, Decision Trees, and K-Nearest Neighbors. Our proposed detection framework is designed to protect the SDN controller using SNORT IDS and IoT devices from DDoS attacks using machine learning models for detecting DDoS attacks. We evaluated our proposed framework using a real smart home testbed utilizing several IoT sensors and software-defined network controlled by a Ryu controller. We evaluated our proposed methods using real data captured from the IoT testbed and standard IoT DDoS attack datasets published. Results showed that ML algorithms can efficiently classify traffic into benign and attack traffic, with the Decision Tree algorithm being able to detect attacks with an accuracy of 99.57%.

While our framework performs well in smart home environments, we recognize the rapidly evolving nature of network security, particularly in IoT settings. To address advanced threats, we plan to explore the integration of modern deep learning methods capable of discerning subtle patterns indicative of sophisticated DDoS attacks. Additionally, we aim to investigate other advanced models for handling fault attacks and apply lightweight cryptographic algorithms to bolster overall security measures. This exploration will contribute to the broader discourse on securing IoT environments against evolving cyber threats.

CRediT authorship contribution statement

Usman Haruna Garba: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Software, Validation, Visualization, Writing – original draft. **Adel N. Toosi:** Funding acquisition, Project administration, Supervision, Writing – review & editing, Methodology, Conceptualization, Formal analysis. **Muhammad Fermi Pasha:** Funding acquisition, Supervision, Writing – review & editing, Resources. **Suleman Khan:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] R. Mahmoud, T. Yousuf, F. Aloul, I. Zualkernan, Internet of Things (IoT) security: Current status, challenges and prospective measures, in: 2015 10th International Conference for Internet Technology and Secured Transactions, ICITST, 2015, pp. 336–341, http://dx.doi.org/10.1109/ICITST.2015.7412116.

[2] T. Mahjabin, Y. Xiao, G. Sun, W.D. Jiang, A survey of distributed denial-of-service attack, prevention, and mitigation techniques, Int. J. Distrib. Sens. Netw. 13 (12) (2017) 33, http://dx.doi.org/10.1177/1550147717741463.

[3] A. Luigi, Samsung devices with support for remote controllers, 2012, https://aluigi.altervista.org/adv/samsux_1-adv.txt.

[4] ENISA, Baseline security recommendation for IoT, 2017, https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot.

[5] C. Kolias, G. Kambourakis, A. Stavrou, J. Voas, DDoS in the IoT: Mirai and other botnets, Computer 50 (7) (2017) 80–84, http://dx.doi.org/10.1109/mc.2017.201.

[6] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J.A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, Y. Zhou, Understanding the mirai botnet, in: 26th USENIX Security Symposium, USENIX Security 17, USENIX Association, Vancouver, BC, 2017, pp. 1093–1110, URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis.

[7] O. Flauzac, C. González, A. Hachani, F. Nolot, SDN based architecture for IoT and improvement of the security, in: 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, 2015, pp. 688–693, http://dx.doi.org/10.1109/WAINA.2015.110.

[8] Incapsula Imperva, DDoS attacks distributed denial of service attack (DDoS) definition, 2011, https://www.incapsula.com/ddos/ddos-attacks.html.

[9] P. Machaka, A. McDonald, F. Nelwamondo, A. Bagula, Using the cumulative sum algorithm against distributed denial of service attacks in Internet of Things, in: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, in: LNICST, vol. 165, 2016, pp. 62–72, http://dx.doi.org/10.1007/978-3-319-29236-6_7.

[10] S. Khan, A. Gani, A.W. Abdul Wahab, M. Guizani, M.K. Khan, Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art, IEEE Commun. Surv. Tutor. 19 (1) (2017) 303–324, http://dx.doi.org/10.1109/COMST.2016.2597193.

[11] Network Working Group, Internet denial-of-service considerations, 2006, https://tools.ietf.org/html/rfc4732.

[12] Cyber Security and Infrastructure Agency, DDoS quick guide, 2020, https://us-cert.cisa.gov/security-publications/DDoS-Quick-Guide.

[13] M. Roesch, Snort: Lightweight intrusion detection for networks, 1999.

[14] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker, P4: programming protocol-independent packet processors, SIGCOMM Comput. Commun. Rev. 44 (3) (2014) 87–95, http://dx.doi.org/10.1145/2656877.2656890.

[15] Freeboard, Freeboard - Visualize the Internet of Things, 2020, http://freeboard.io/.

[16] Xerxes, Xerxes - Dos tool, 2020, https://github.com/sepehrdaddev/Xerxes.

[17] Hping3, hping3 package description, 2020, https://tools.kali.org/information-gathering/hping3.

[18] Radware, DDoSPedia index, 2021, https://www.radware.com/security/ddos-knowledge-center/ddospedia/.

[19] TensorFlow, An end-to-end open source machine learning platform, 2020, https://www.tensorflow.org/.

[20] I. Sharafaldin, A.H. Lashkari, S. Hakak, A.A. Ghorbani, Developing realistic distributed denial of service (ddos) attack dataset and taxonomy, in: 2019 International Carnahan Conference on Security Technology, ICCST, 2019, pp. 1–8, http://dx.doi.org/10.1109/CCST.2019.8888419.

[21] N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull, Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset, Future Gener. Comput. Syst. 100 (2019) 779–796, http://dx.doi.org/10.1016/j.future.2019.05.041.

[22] F.S.D.L. Filho, F.A.F. Silveira, A. de Medeiros Brito Júnior, G. Vargas-Solar, L.F.Q. Silveira, Smart detection: An online approach for DoS/DDoS attack detection using machine learning, Secur. Commun. Netw. 2019 (2019) 1574749:1–1574749:15, http://dx.doi.org/10.1155/2019/1574749.

[23] S.M. Tahsien, H. Karimipour, P. Spachos, Machine learning based solutions for security of Internet of Things (IoT): A survey, J. Netw. Comput. Appl. 161 (February) (2020) http://dx.doi.org/10.1016/j.jnca.2020.102630, arXiv:2004.05289.

[24] J. Zurawski, J.M. Schopf, National Institute of Standards and Technology Requirements (Analysis Report), Technical Report, Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2023.

[25] M. El-Hajj, H. Mousawi, A. Fadlallah, Analysis of lightweight cryptographic algorithms on IoT hardware platform, Future Internet 15 (2) (2023) 54, http://dx.doi.org/10.3390/fi15020054.

[26] Y. Wang, L. Liu, C. Si, B. Sun, A novel approach for countering application layer DDoS attacks, in: 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference, IAEAC, 2017, pp. 1814–1817, http://dx.doi.org/10.1109/IAEAC.2017.8054326.

[27] Y. Xie, S.-Z. Yu, Monitoring the application-layer DDoS attacks for popular websites, IEEE/ACM Trans. Netw. 17 (1) (2009) 15–25, http://dx.doi.org/10.1109/TNET.2008.925628.

[28] S. Ranjan, R. Swaminathan, M. Uysal, A. Nucci, E. Knightly, DDoS-shield: DDoS-resilient scheduling to counter application layer attacks, IEEE/ACM Trans. Netw. 17 (2009) 26–39, http://dx.doi.org/10.1109/TNET.2008.926503.

[29] R. Braga, E. Mota, A. Passito, Lightweight DDoS flooding attack detection using NOX/OpenFlow, in: IEEE Local Computer Network Conference, 2010, pp. 408–415, http://dx.doi.org/10.1109/LCN.2010.5735752.

[30] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1–6, http://dx.doi.org/10.1109/CISDA.2009.5356528.

[31] P. Kasinathan, C. Pastrone, M. Spirito, M. Vinkovits, Denial-of-service detection in 6lowpan based internet of things, in: International Conference on Wireless and Mobile Computing, Networking and Communications, 2013, pp. 600–607, http://dx.doi.org/10.1109/WiMOB.2013.6673419.

[32] M. Özçelik, N. Chalabianloo, G. Gür, Software-defined edge defense against IoT-based ddos, in: 2017 IEEE International Conference on Computer and Information Technology, CIT, 2017, pp. 308–313, http://dx.doi.org/10.1109/CIT.2017.61.

[33] J. Bakker, B. Ng, W. Seah, Can machine learning techniques be effectively used in real networks against DDoS attacks? 2018, pp. 1–6, http://dx.doi.org/10.1109/ICCCN.2018.8487445.

[34] M. Nobakht, V. Sivaraman, R. Boreli, IEEE, A host-based intrusion detection and mitigation framework for smart home IoT using OpenFlow, in: Proceedings of 2016 11th International Conference on Availability, Reliability and Security, Ares 2016, 2016, pp. 147–156, http://dx.doi.org/10.1109/ares.2016.64.

[35] K. Sonar, H. Upadhyay, An approach to secure internet of things against DDoS, ISBN: 978-981-10-0133-8, 2016, pp. 367–376, http://dx.doi.org/10.1007/978-981-10-0135-2_36.

[36] S.S. Bhunia, M. Gurusamy, IEEE, Dynamic attack detection and mitigation in IoT using SDN, in: 2017 27th International Telecommunication Networks and Applications Conference, 2017, pp. 84–89.

[37] R. Doshi, N. Apthorpe, N. Feamster, Machine learning DDoS detection for consumer internet of things devices, in: 2018 IEEE Security and Privacy Workshops, SPW, 2018, pp. 29–35, http://dx.doi.org/10.1109/SPW.2018.00013.

[38] H. Gordon, C. Batula, B. Tushir, B. Dezfouli, Y. Liu, Securing smart homes via software-defined networking and low-cost traffic classification, in: 2021 IEEE 45th Annual Computers, Software, and Applications Conference, COMPSAC, IEEE, 2021, pp. 1049–1057.

[39] P. Binu, D. Mohan, E.S. Haridas, An sdn-based prototype for dynamic detection and mitigation of dos attacks in iot, in: 2021 Third International Conference on Inventive Research in Computing Applications, ICIRCA, IEEE, 2021, pp. 5–10.

[40] H.M. Mohammad, A.A. Abdullah, DDoS attack mitigation using entropy in SDN-IoT environment, in: AIP Conference Proceedings, vol. 2591, (no. 1) AIP Publishing LLC, 2023, 020002.

[41] N. Sambandam, M. Hussein, N. Siddiqi, C. Lung, Network security for IoT using SDN: Timely DDoS detection, in: 2018 IEEE Conference on Dependable and Secure Computing, DSC, 2018, pp. 1–2, http://dx.doi.org/10.1109/DESEC.2018.8625119.

[42] POX, POX controller, 2018, https://noxrepo.github.io/pox-doc/html/.

[43] N. Ravi, S.M. Shalinie, Learning-driven detection and mitigation of DDoS attack in IoT via SDN-cloud architecture, IEEE Internet Things J. 7 (4) (2020) 3559–3570, http://dx.doi.org/10.1109/JIOT.2020.2973176.

[44] M.M. Cherian, S.L. Varma, Mitigation of DDOS and MiTM attacks using belief based secure correlation approach in SDN-based IoT networks, Int. J. Comput. Netw. Inf. Secur. 14 (1) (2022) http://dx.doi.org/10.5815/ijcnis.2022.01.05.

[45] W.I. Khedr, A.E. Gouda, E.R. Mohamed, FMDADM: A multi-layer ddos attack detection and mitigation framework using machine learning for stateful SDN-based IoT networks, IEEE Access 11 (2023) 28934–28954, http://dx.doi.org/10.1109/ACCESS.2023.3260256.

[46] N. Patel, B. Mehtre, R. Wankar, A snort-based secure edge router for smart home, Int. J. Sens. Netw. 41 (1) (2023) 42–59.

[47] M.A. Bouke, A. Abdullah, S.H. ALshatebi, M.T. Abdullah, H. El Atigh, An intelligent DDoS attack detection tree-based model using Gini index feature selection method, Microprocess. Microsyst. 98 (2023) 104823, http://dx.doi.org/10.1016/j.micpro.2023.104823.

[48] P. Chauhan, M. Atulkar, A framework for DDoS attack detection in SDN-based IoT using hybrid classifier, in: Machine Learning, Image Processing, Network Security and Data Sciences: Select Proceedings of 3rd International Conference on MIND 2021, Springer, 2023, pp. 889–900.

[49] D. Yin, L. Zhang, K. Yang, A DDoS attack detection and mitigation with software-defined Internet of Things framework, IEEE Access 6 (2018) 1, http://dx.doi.org/10.1109/ACCESS.2018.2831284.

[50] J. Galeano-Brajones, J. Carmona-Murillo, J.F. Valenzuela-Valdés, F. Luna-Valero, Detection and mitigation of DoS and DDoS attacks in iot-based stateful SDN: An experimental approach, Sensors (Switzerland) 20 (3) (2020) 1–18, http://dx.doi.org/10.3390/s20030816.

[51] M. Ozcelik, N. Chalabianloo, G. Gur, IEEE, Software-defined edge defense against IoT-based DDoS, in: 2017 IEEE International Conference on Computer and Information Technology, 2017, pp. 308–313, http://dx.doi.org/10.1109/cit.2017.61.

[52] M. Hasan, M.M. Islam, M.I.I. Zarif, M. Hashem, Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches, Internet Things 7 (2019) 100059, http://dx.doi.org/10.1016/j.iot.2019.100059.

[53] P. Manso, J. Moura, C. Serrão, SDN-based intrusion detection system for early detection and mitigation of DDoS attacks, Information 10 (3) (2019) 106, http://dx.doi.org/10.3390/info10030106.

[54] E. Anthi, L. Williams, P. Burnap, Pulse: An adaptive intrusion detection for the Internet of Things, in: Living in the Internet of Things: Cybersecurity of the IoT - 2018, 2018, pp. 1–4, http://dx.doi.org/10.1049/cp.2018.0035.

[55] Kamaldeep, M. Malik, M. Dutta, Contiki-based mitigation of UDP flooding attacks in the internet of things, in: 2017 IEEE International Conference on Computing, Communication and Automation, 2017, pp. 1296–1300.

[56] J. Samandari, C. Gritti, Post-quantum authentication in the MQTT protocol, J. Cybersecur. Priv. 3 (3) (2023) 416–434, http://dx.doi.org/10.3390/jcp3030021.

[57] M. Anastasova, R. El Khatib, A. Laclaustra, R. Azarderakhsh, M.M. Kermani, Highly optimized Curve448 and Ed448 design in wolfSSL and side-channel evaluation on cortex-M4, in: 2023 IEEE Conference on Dependable and Secure Computing, DSC, IEEE, 2023, pp. 1–8.

[58] R. Elkhatib, B. Koziel, R. Azarderakhsh, M.M. Kermani, Cryptographic engineering a fast and efficient SIKE in FPGA, ACM Trans. Embed. Comput. Syst. (2023) http://dx.doi.org/10.1145/3584919.

[59] S. Li, Y. Chen, L. Chen, J. Liao, C. Kuang, K. Li, W. Liang, N. Xiong, Post-quantum security: Opportunities and challenges, Sensors 23 (21) (2023) 8744, http://dx.doi.org/10.3390/s23218744.

[60] J. Kaur, A.C. Canto, M.M. Kermani, R. Azarderakhsh, Hardware constructions for error detection in WG-29 stream cipher benchmarked on FPGA, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2023) http://dx.doi.org/10.1109/TCAD.2023.3338108.

[61] J. Smith, A. Johnson, Block cipher QARMA with error detection mechanisms, in: Proceedings of the IEEE International Conference on Cryptography, London, UK, 2023, pp. 29–30.

[62] N.H. Noordin, P.S. Eu, Z. Ibrahim, FPGA implementation of metaheuristic optimization algorithm, e-Prime-Adv. Electr. Eng. Electron. Energy 6 (2023) 100377, http://dx.doi.org/10.1016/j.prime.2023.100377.

[63] K.D. Muthavhine, M. Sumbwanyambe, An application of the Khumbelo function on the Camellia algorithm to prevent attacks in IoT devices, IEEE Access 11 (2023) 119959–119992, http://dx.doi.org/10.1109/ACCESS.2023.3312789.

[64] C. An, W. Bai, D. Zhang, Meet-in-the-middle differential fault analysis on Midori, Electron. Res. Arch. 31 (11) (2023) 6820–6832.