IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 30 June 2025, accepted 27 July 2025, date of publication 31 July 2025, date of current version 7 August 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3594580

RESEARCH ARTICLE

IoTShield: Defending IoT Systems Against Prevalent Attacks Using Programmable Networks

MAH-RUKH FIDA[1,*], AZZA H. AHMED[2,*], AND AMEER SHAKAYB ARSALAAN[3], (Member, IEEE)

[1]School of Business, Computing and Social Sciences, University of Gloucestershire, GL50 2RH Cheltenham, U.K.
[2]Simula Metropolitan Center for Digital Engineering, 0167 Oslo, Norway
[3]Department of Computer Science, University of Swabi, Swabi, KP 23340, Pakistan

Corresponding author: Azza H. Ahmed (azza@simula.no)

This work is supported by QR funding provided by the University of Gloucestershire, UK.

Mah-Rukh Fida and Azza H. Ahmed contributed equally to this work.

ABSTRACT The growing proliferation of Internet of Things (IoT) devices in smart homes, smart agriculture, and smart energy grids has greatly improved their functionality, efficiency, and responsiveness — but it has also widened the attack surface of these networks. The inherent security vulnerabilities of IoT devices, have rendered them susceptible to a variety of flow-based attacks such as Distributed Denial of Service (DDoS), scanning, spoofing, data exfiltration and web-based attacks, thereby diminishing their potential benefits. This paper presents `IoTShield`, a Software Defined Network (SDN) based dual-stage defensive framework, designed to mitigate different flow-based attacks targeting IoT systems. Leveraging recent advancements in programmable networks, our defensive framework enables each programmable switch within the *connectivity layer* of the network to be responsible of identifying a single attack category among prevalent attacks. Furthermore, to effectively mitigate the spread of these attacks, detected attacks are classified at the network controller, facilitating timely updates to the data plane defensive rules. As a proof of concept, using CICIoT2023 dataset, we first illustrate that deploying separate detectors for DDoS and Web-based attack categories on programmable data planes reduces *false alarms* by 58% and 97%, respectively. Furthermore, a single DDoS attacks detector based on lightweight Decision Tree (DT) model in the data plane, achieves 80-99% of accuracy in detecting different types of attack flows, with fine-grained classification offloaded to the control plane where a Convolutional Neural Network (CNN) classifier achieves 99% accuracy. Besides, IoTShield significantly reduces the latency and load on controller to perform the attack detection; with only 0.14 milliseconds of additional median queuing delay.

INDEX TERMS Software defined network, programmable networks, DDoS attacks, IoT, in-network machine learning.

I. INTRODUCTION

The Internet of Things (IoT) has emerged as a significant network that encompasses a vast array of devices interconnected to streamline various societal activities such as home, healthcare, transportation, education, aviation, and industries. These devices range from minuscule sensors to sizable gadgets like actuators, mobile phones, televisions, light bulbs, thermostats, medical equipment, smartwatches, software, and more. The global count of IoT devices is projected to nearly double, surging from 15.1 billion in 2020 to over 29 billion by 2030, offering convenience, efficiency, and connectivity across various sectors [1]. However, despite the potential benefits, challenges such as interoperability issues among diverse devices and protocols, along with limited security capabilities, pose significant hurdles to safeguarding IoT networks. These challenges create opportunities for attackers to exploit vulnerabilities within the system.

The growing diversity of attacks complicates the privacy and security of IoT devices and their networks. Attacks such as DDoS [2], Web-based attacks [3], Bruteforce and Scanning attacks [4], Spoofing, Replay, and Man-In-The-Middle attacks are increasingly targeting IoT networks. These attacks exhibit varying patterns, making

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Quan.

a one-size-fits-all solution ineffective. To address the challenges posed by the increasing number of heterogeneous IoT devices and the large volume of generated traffic, network vendors are transitioning to Software Defined Networking (SDN) [5], [6]. SDN facilitates the virtualization of networks and services, enabling swift and automated reconfiguration of network devices, traffic rerouting, implementation of new access rules, and more efficient management of overloaded networks. This paper, therefore, introduces an SDN based framework, under the control of a single service provider, to address diverse attacks within an IoT network.

We present a dual-stage distributed defensive system for attack detection and classification, which in turn paves the way for its mitigation. The data plane of the SDN system is composed of programmable switches, some of which are designated for attack detection. Each switch among the designated switches is responsible for triggering flows that fall under same attack category. Tailoring a switch pipeline, to track similar attacks patterns not only reduces monitoring overhead but also enhances detection accuracy of an attack category. To ensure a lightweight defensive mechanism, each switch sends samples from the anticipated malicious flows to the controller in the SDN, to classify the type of attack within the attack category and update rules on the switch(s), so that to block the propagation of similar flows in the network. Our contributions in this study are therefore, as follows:

- We present `IoTShield`; a dual-stage framework to mitigate the propagation of prevalent attack flows within IoT network. Administered by same SDN, a set of data plane switches are designated as attack detectors, with each responsible for triggering malicious flows with similar pattern.
- As a proof of concept, we deploy a programmable software switch within a Mininet environment. Initially, this setup is used to demonstrate that a single programmable switch should ideally handle only one attack category. To validate this, we compare the performance of two separate tree-based detector models—each dedicated to detecting either DDoS or web-based attacks—against a single, combined tree-based detector. The results show that using separate detectors for similar attack patterns yields significantly better accuracy and reduces processing load. Building on this, we proceed to the next phase, where a single detector deployed on the programmable switch is tasked with identifying four different types of DDoS attacks.
- For the detection of DDoS attack category, we leverage a lightweight Decision Tree with depth of 10, accurately flagging 80-99% of DDoS flows. On the other hand, its false flag reporting is less than 3%.
- Selective packets of marked malicious flows, with payloads being truncated, are sent to the SDN controller that uses Convolution Neural Network (CNN) model to classify the type of the attack. For DDoS, the classification achieves 99% accuracy. This is done to update the defensive rules in the network switches so that to halt the propagation of the attack flows at the earliest opportunity.
- Delving deeper into the overhead and efficiency brought by `IoTShield`, we find that with 0.14 *ms* increase in median queuing delay, 4% increase on CPU load and up to 14% increase in memory utilization of the software switches, the dual-stage defensive framework reduces the link load by up to 99.8%.

The paper comprises seven additional sections. Following the description of the study's motivation in section II, section III presents the `IoTShield` framework for attack detection and classification. Section IV provides a comprehensive overview of the data plane detector module, control plane attack classifier module, and the experimental setup. In section V, a thorough analysis of the attack detection and classification process within the dual-stage framework is presented, alongside an assessment of the performance enhancements and limitiations of `IoTShield` in section VI. Subsequently, a review of related literature is provided in section VII before concluding the paper with insights into future research directions in section VIII.

II. MOTIVATION

The rapid expansion of the IoT has transformed industries by interconnecting billions of devices. However, this proliferation has brought significant security challenges. A. Petrosyan [7] reports a sharp increase in IoT cyberattacks, rising from 32 million in 2018 to 112 million in 2022. A Zscaler report from November 2024 highlights a 400% surge in IoT-targeted attacks in recent years. Similarly, the 2024 IoT security report by NetGear/Bitdefense notes that smart home devices now endure up to 10 attacks per day [8]. This increase is not only in volume but also in the diversity of attack vectors [9], [10], placing greater demands on the scalability and robustness of security infrastructure.

Traditional network security mechanisms—such as firewalls and signature-based Intrusion Detection System (IDS)—rely on static rules and are often ineffective against the dynamic and flow-based nature of IoT threats. These legacy solutions typically lack stateful inspection capabilities and fail to detect emerging or polymorphic threats. Moreover, they struggle with scalability and are unable to adapt to the heterogeneous and ever-changing IoT landscape.

Compounding the problem, many IoT devices are resource-constrained and lack native security features. This leaves the broader network vulnerable to attacks such as DDoS, spoofing, and data interception. Additionally, IoT deployments often span across multiple domains (e.g., smart homes, healthcare, industrial systems), where fragmented security enforcement and lack of centralized visibility hinder efficient threat detection and real-time response.

SDN offers a promising solution to these challenges by decoupling the control and data planes, allowing for centralized and programmable network management.

SDN enhances network visibility, enables dynamic policy enforcement, and supports the integration of intelligent threat detection systems using Machine Learning (ML) for anomaly detection and automated mitigation. SDN has also been advocated as a scalable framework to manage communication among IoT gateways, cloud servers, and a multitude of sensors and devices [11]. In an SDN-enabled IoT environment, the network architecture typically comprises four layers [12] (see Figure 1):

- *Device Layer*: This foundational layer includes various endpoint devices such as RFID tags, smartphones, smart vehicles, thermostats, cameras, and other sensing devices along with actuators.
- *Connectivity Layer:* Devices connect to the network through wireless access points, base stations, and gateways, that inter-turn connect directly or via switches and routers forming a link to the broadband backbone. In SDN-IoT architecture, these nodes serve primarily as data transmission points, deferring control decisions to the centralized SDN controller.
- *Control Layer*: Serving as the intermediary between the application and connectivity layer, this layer is where the SDN controller resides. It facilitates programmable management through south-bound Application Programming Interfaces (APIs), allowing developers to enforce policies and allocate hardware resources across different services.
- *Application Layer*: This top layer delivers essential services to the IoT network, including mobility management, security, network monitoring, and IoT application-specific functionalities. Through northbound APIs, it interfaces with the control layer, facilitating real-time optimization and management of IoT applications.

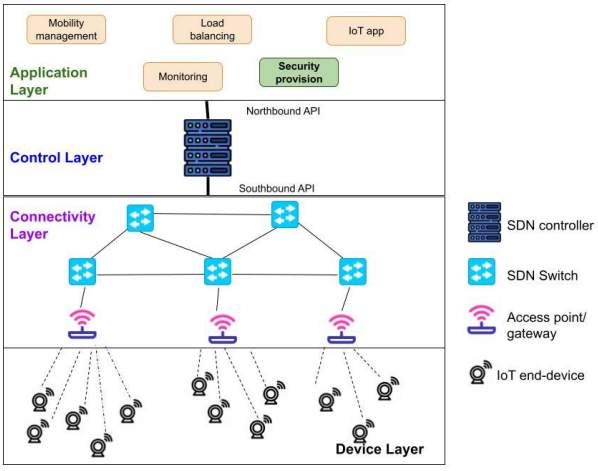

FIGURE 1. Layers of an SDN-based IoT network.

As a southbound API, OpenFlow [13] is a widely deployed protocol. It is responsible for the communication between forwarding devices and the software-based controller in the SDN's architecture. With respect to security services, OpenFlow-based switches collect samples of data packets and transmit it to the SDN controller for traffic analysis. This not only puts additional overhead on the controller, impacting its performance as well as scalability, it is difficult to achieve fine identification and comprehensive defense with sampling instead of packet-by-packet inspection.

With programmable switch ASICs and SmartNIC, the forwarding data plane has more freedom to take adaptive decisions and act accordingly. These programmable devices employ glsp4, a hardware-independent programmable language with line-speed processing capabilities of up to hundreds of gigabytes. The programmable data plane has evidenced a break-through in handling not only quality of services, traffic engineering, load balancing [14], [15], [16] but also network security issues [17]. With the support of customized packet processing, Programming Protocol-independent Packet Processor (P4) can provide defenses against different flow-based attacks such as Address Resolution Protocol (ARP) attacks, DDoS attacks, Web-based attacks, Eavesdropping [18], Replay attacks and Spoofing attacks.

There are three key challenges when employing in-network attack mitigation mechanisms. First, programmable data planes have limited on-chip memory, such as TCAM and SRAM, which restricts their capacity to support complex processing tasks. Second, due to limitations of programmable devices, the programming language P4 does not support floating-point operations, which prevents it from performing calculations involving division. exponential, or logarithmic functions. Third, the flow characteristics of different types of network attacks vary significantly, making it impractical to implement a unified attack mitigation model on a single switch—especially when stateful, flow-level features are required. Prior research shows that even within a single attack category like DDoS, programmable data plane-based detection and mitigation can only address specific subtypes at a time. For example, C.M. Iurian [19] developed a P4-based detector for SYN flooding attacks alone, while S.Z. Yang [20] proposed separate detectors for SYN and UDP flooding attacks, respectively. In summary:

- A single programmable switch cannot support holistic detection of diverse flow-based attacks due to resource constraints. For instance, the memory capacity of a typical switch is only around 50MB [21], and deploying multiple match-action pipelines significantly degrades the throughput of benign traffic.
- Existing literature does not provide a systematic approach for attack-type classification, which is essential for effective mitigation. For instance, to counter a SYN flood, detection logic must identify when the number of SYN packets in a TCP flow exceeds a certain threshold T compared to ACK packets, as proposed by Poseidon [22], and then enforce drop rules accordingly. Similarly, TCP flood attacks can be mitigated via rate-limiting rules. Such compact, targeted rule sets bypass

lengthy detection pipelines, reducing propagation of malicious traffic within the network.
- Since it is rare for multiple types of attacks to occur simultaneously, mitigation logic should be dynamically updated based on real-time classification. This enables efficient defense with minimal processing overhead at the switch level.

To address these challenges, this paper presents a distributed two-stage framework for detecting and attributing various attack types. While `IoTShield` is particularly well-suited for IoT networks—due to their limited computational capacity, reliance on default credentials, and infrequent updates that heighten their vulnerability—it is equally applicable to data centers and other networks with centralized control and visibility through SDN controllers. In IoT settings, attacks can be effectively mitigated closer to the device layer by enabling early detection and suppression at edge switches, thereby enhancing response time and reducing impact.

Although Figure 1 illustrates a single SDN domain, the architecture supports collaborative intelligence across multiple SDN domains. Inference models can be trained in a decentralized manner—similar to Federated Learning—allowing the exchange of threat intelligence and model updates via east-west interfaces without centralizing sensitive data.

III. `IoTSHIELD` SYSTEM ARCHITECTURE

To address the detection and attribution of various attacks on an IoT network, we start by outlining the essential requirements of the defensive architecture in the subsequent subsection. Following this, we delve into the components of the system and illustrate their functionality.

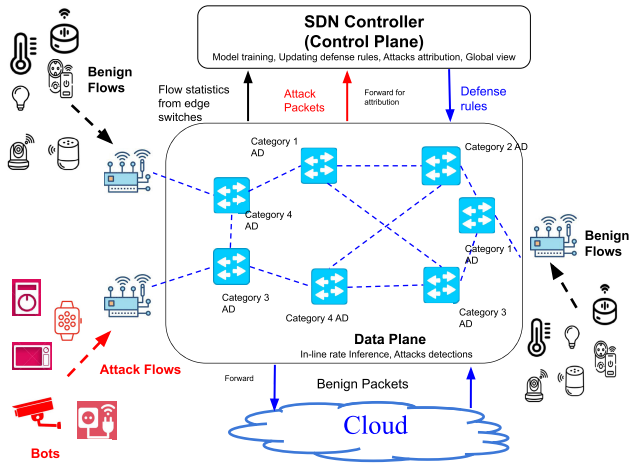

FIGURE 2. `IoTShield` **Architecture: with programmable data-plane responsible for *Attack Detection* and control-plane for *Attributing* the type of the malicious flows.**

A. KEY REQUIREMENTS

A high performance defensive architecture has several requirements:

1) High accuracy: The architecture must demonstrate high performance in terms of accuracy. The system should achieve high detection rates for IoT attacks while minimizing false alarms.
2) Line-rate processing: Programmable switches are typically compact devices with significant constraints in both memory and processing power [23]. Since their primary role is packet forwarding and routing, security functions are considered auxiliary and must not interfere with line-rate performance. To maintain this high-speed processing, any offloaded ML model must be lightweight, with inference times kept minimal to ensure timely detection of malicious flows without degrading throughput.
3) Restricted computational capabilities: In addition to constraints such as limited memory and a small number of processing stages, programmable network devices also lack floating-point support. This limitation means that P4 does not support advanced mathematical operations such as trigonometric functions, exponentiation, or complex control structures beyond`if-else` statements. It is primarily limited to basic operations like integer addition/subtraction and bit shifts, and does not facilitate floating-point arithmetic. Additionally, it does not support loops, recursions and dynamic memory allocation. These constraints pose challenges for deploying complex models like deep learning neural networks, as any offloaded model must operate within the confines of these limitations.

B. TWO-LEVEL INFERENCE

To fulfill the outlined requirements, we introduce `IoTShield`, a defensive architecture illustrated in Figure 2. In essence, `IoTShield` executes two primary functions: *detecting* IoT attacks through a distributed data plane and then *attributing* their nature via the control plane. The subsequent sections delve into the description of each component.

1) DETECTION AT DATA PLANE

Our objective here is to optimize the performance of ML inference while ensuring the model remains compatible with the ingress pipeline of the P4 switch. Although the realm of ML offers numerous supervised learning techniques, not all are suitable for our specific task. In addition to standard ML model requirements like accuracy, our focus lies on developing a model that is both fast and lightweight in terms of memory usage. To achieve this, within our framework, we've opted for a tree-based model. This choice is driven by our aim to strike a balance between speed and memory efficiency. While tree-based models excel at handling non-linear problems, they might not be as powerful as more complex methods, such as neural networks with multiple hidden layers. Despite this, a tree-based model can achieve acceptable accuracy levels and often work faster than neural networks.

It is important to highlight that the detection accuracy of tree-based models typically improves with an increase in the number and depth of trees, albeit at the cost of longer computational time. Consequently, our approach focuses on minimizing both the number of trees and their depth while still achieving a satisfactory level of accuracy.

2) ATTRIBUTION AT CONTROL PLANE

The SDN controller plays several critical roles within the `IoTShield` architecture. Firstly, it is responsible for training and deploying the attack detection model onto the data plane. Training an ML model is resource-intensive in terms of both CPU and memory usage. The training process involves the controller polling the edge switches for recent flow statistics at an adaptive rate, as detailed in [24]. Using Deep Packet Inspection (DPI), if a flow pattern changes—regardless of whether it is benign or represents a new type of malicious traffic—the detection module(s) are updated accordingly.

Additionally, we capitalize on the higher computational capabilities of the SDN controller by deploying an efficient and advanced attacks classifier to *attribute* the nature of the malicious flow/type of attacks once detected by the data plane. We opt for a CNN model for this task, given its successful application across various domains and its high performance in extracting features from network traffic data [25].

Lastly, with global visibility of the control plane, we assume it is aware of flow paths and the locations of programmable switches within the network domain. It distributes and assigns different detectors to various switches so that traffic is analyzed for prevalent attacks throughout its flow duration, thereby reducing the likelihood of any traffic flow going unnoticed.

Following the assessment of the attack type inflicted by the flow, intrusion mitigation rules are updated on the data plane accordingly.

C. DISTRIBUTED DETECTION AT DATA PLANE

With inherent challenges of the IoT devices, a variety of attacks are launched that fall in broader categories of e.g., DDoS attacks, Reconnaissance attacks, Web-based attacks and Spoofing attacks, to name a few. To ensure security and integrity of the IoT network, a holistic defense mechanism is desirable. With the assumption that multiple programmable switches fall within the end-to-end path of IoT data traffic, deploying a complex ML model on a single switch i.e., ingress switch can burden both the limited processing and memory resources of a switch. Additionally, it increases false alarms and misses attacks due to differences among the patterns of the flows belonging to different attack categories.

To illustrate this, we take four different attack types within DDoS category and three different types of attacks falling with Web-based attack category, along with a benign dataset drawn from CICIoT2023 [26], the size and details of which

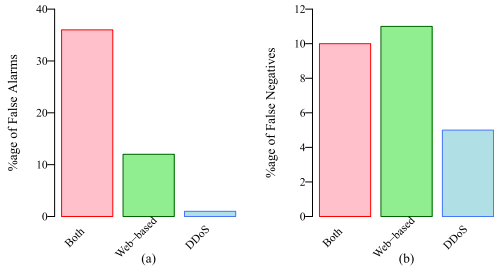

FIGURE 3. False alarm and false negatives, with a DT detector trained for both DDoS and Web-based attacks, compared to separate detectors for each attack category.

are given in Table 1. Figure 3 (a), depicts that when an attack detector based-on a DT model is employed in a P4 software switch, to differentiate *benign* flows from the *attack* flows comprising DDoS and Web-based attacks, 37% of the *benign* packets are wrongly triggered as *attack*. On the other hand, training separate detectors for the two attack categories, the false alarms drop by 58% and 97%, respectively. It is further observed, in Figure 3 (b), that compared to a separate DDoS detector, the accuracy of combined detector drops. Although the poor accuracy of the combined detector partly seems to be due to Web-based attack patterns, that seem to be similar in some way to the benign pattern as is indicated by higher number of misses by the separate Web-based attacks detector. It is worth noting that rather than using separate switches for detecting different categories of attack, these can be better used for sub-categories that follow similar flow pattern. Such clustering however, needs in-depth investigation of the prevalent attack types on a network.

In essence, Figure 2illustrates a two-level inference and distributed detection framework. The first level operates at the data plane, where switches within the IoT network are assigned to detect specific categories of attacks. Each switch is equipped with a dedicated inference module tailored to a particular threat type (e.g., DDoS, spoofing), enabling decentralized, near-source detection as traffic flows through the network. When a threat is flagged, the SDN controller performs second-level inference by attributing the attack more precisely, updating mitigation rules on the relevant switch, and conducting proactive DPI-assisted analysis using features periodically collected from the network. By combining this proactive analysis with a global view, the controller strategically places detection modules—deploying them at edge switches for frequent threats to enable early containment, and deeper in the network for less common or more complex attacks—thus ensuring efficient, context-aware protection across the IoT environment.

IV. IMPLEMENTATION

In this section we develop a prototype of the proposed architecture. We describe the implementation details of the essential elements of `IoTShield`, including the P4 data plane module, the control plane, the ML models, and the IoT attacks dataset.

TABLE 1. Types of attacks considered in two different attack category.

Type	Records	Tool
Benign	54246	–
DDoS Attack		
Ack-Fragmentation	20974	hping3 (29)
RSTFIN Flood	30162	hping3 (29)
TCP Flood	32653	hping3 (29)
SYN Flood	48760	hping3 (29)
Web-based Attack		
Cross-site Scripting	25432	DVWA (30)
SQL Injection	36763	DVWA (30)
Command Injection	40857	DVWA (30)

A. P4 DATA PLANE MODULE

In the data plane, we employ the P4 language to execute the feature extractor and online detection module on a BMv2 software switch. To implement the detection module we use the `if-else` programming construct that fits well for a look-up of conditions on features specified, at the nodes of a tree-based model, akin to the methodology utilized in pForest [27] and SwithcTree [28].

Tree-based Inference using P4. A tree-based model e.g., a Random Forest (RF) model may comprise multiple DTs, each providing a label for a given packet sample. Within an RF a single DT can be conceptually linked to a match-action pipeline in P4. At each stage of the pipeline, conditions are applied to the features, directing the flow through different branches of the tree. The number of stages in the pipeline corresponds to the number of extracted features. The final stage categorizes packet types as either *benign* or *malicious*. In our study, we focus on an RF attack detector with a minimal number of trees and tree-depth, while still achieving reasonable accuracy.

B. CONTROL PLANE MODULE

The controller within the SDN framework acts as a training environment for the ML model deployed in the data plane. Through Python, we not only train the ML-based detector models, but also utilize them to translate the RF tree(s) into P4 match-action pipelines. Furthermore, the computational capabilities of the controller are harnessed for training and attributing the *malicious* flows detected by the data plane.

We employ a CNN for attribution purposes. Our model comprises three convolutional layers responsible for feature extraction from the input data. The output of each convolutional layer is passed through a Rectified Linear Unit (ReLU) activation function. Following each convolutional layer, a pooling layer, typically employing max pooling, is applied to reduce the spatial dimensions of the feature maps while preserving essential information. Ultimately, a softmax classifier is employed on the output of the last fully connected layer to determine the probabilities of the input belonging to each class in the classification task. During the model training process, a loss function is defined based on cross-entropy and run for 50 epochs.

C. IoT ATTACKS DATASET

To develop and assess our system, we employed the CICIoT2023 [26] dataset. This dataset comprises PCAP files and their corresponding CSV versions capturing 33 distinct attacks conducted within an IoT environment consisting of 105 devices. These devices encompass various types such as smart home devices, cameras, sensors, and microcontrollers. The attacks are categorized into seven types: DDoS (12 sub-types), DoS (4 sub-types), Recon (5 sub-types), Web-based (6 sub-types), Brute force (1 sub-types), Spoofing (2 sub-types), and Mirai (3 sub-types)

As part of our proof of concept, we investigate multiple subtypes within two primary categories: DDoS and web-based attacks. The dataset comprises approximately 289,000 records, including 54,200 *benign* entries. Each attack type's CSV file is generated from PCAP files of at least 5MB in size. The distribution of records across different attack types is summarized in Table 1.

Within these categories, we evaluate the performance of both combined and separate detection models, as illustrated in Figure 3. Furthermore, we implement a P4-based detector and a controller-based attributor specifically designed for the DDoS category. Given the prevalence and complexity of DDoS attacks [31], our focus is directed toward this category, particularly on the following four attack types.

1) A *fragmented ACK flood attack* uses a relatively small number of packets of maximum size (e.g., 1500 bytes) to fill bandwidth. In most cases, fragmented ACK packets easily pass through routers, access control lists (ACLs), firewalls, and intrusion prevention systems because these devices do not reassemble fragmented packets at the network level. Such packets typically contain random data. As the attacker aims to fill the entire bandwidth of the victim's external network channels, this type of flood attack degrades the performance of all servers in the targeted network.
2) During a *RST* or *FIN flood*, the victim server receives spoofed RST or FIN packets at high speed that are unrelated to any of the sessions in the server database. The victim server is forced to allocate a significant amount of system resources to match incoming packets with current connections, resulting in degraded server performance and partial inaccessibility.
3) *TCP flood* establishes a large number of connections without transferring data, or slowly transferring data for the purpose of exhausting the resources of the victim's TCP stack.
4) *SYN flood* is based on the TCP three-way handshake algorithm. The malicious entity rapidly sends server connection requests containing a spoofed source IP address. SYN flood is gradually taking up all memory of the connection table.

D. FEATURES EXTRACTION & ENGINEERING

As detailed in Section III, our decision to deploy a RF as a lightweight detector prioritizes minimizing the number

of features. Unlike the approach in [26], we refrain from incorporating features that might cause model failure when new attack flows, not closely related in time, emerge within the network. For instance, [26] utilizes flow ID, comprising <source IP, destination IP, source port, destination port number, transport layer protocol>, and packet generation timestamps from the PCAP files. Due to the argument that similar attack types can occur for any other <source IP, destination IP> pair at any time, we exclude these features. Additionally, we avoid including features that are currently non-computable by P4 version 16, such as those representing the variance and standard deviation of feature values for the flows.

For simplicity, we restrict P4 parsing to Ethernet, IPv4, and TCP/UDP headers. We then follow traditional procedures for feature selection by initially training an RF model with all available features and obtaining a feature ranking based on the Mean Decrease in Impurity (MDI).

Subsequently, we train additional models by sequentially adding features to determine the smallest subset capable of achieving performance comparable to or better than that of the full model. The considered features along with their importance based on MDI are listed in Table 2.

To train the CNN model for attributing the detected DDoS attack, we use features shaded in Table 2 alongside TCP header flags, namely, `fin_flag_number`, `syn_flag_number`, `psh_flag_number`, `ack_flag_number`, `urg_flag_number`, `ece_flag_number`, and `cwr_flag_number`. These features track the count of specific type of flag seen till a packet per flow. We also include `srcport` and `dstport` which show improvement in the model accuracy.

E. EXPERIMENTAL SETUP

To illustrate the proposed concept, we utilize a Mininet-based network setup shown in Figure 4. Although the simulated topology is deliberately simplified, the evaluation is grounded in realistic traffic traces from CICIoT2023, allowing us to assess the trade-off between detection accuracy and system load under varied traffic conditions. The primary objective is to demonstrate the feasibility and effectiveness of the two-level attack attribution framework.

All three switches in Figure 4 operate as P4 BMv2 switches using the V1Model architecture. A Python script is used to send PCAP packets from the sending host to Switch 1, which runs a P4 16 program implementing match-action rules derived from the previously discussed tree-based model. The internal workflow of Switch 1 is illustrated in Figure 5. Upon receiving a packet, Switch 1 parses its headers and extracts packet-level features such as source and destination IP addresses, SYN flag status, and more. If the mitigation rules match at the ingress pipeline, appropriate actions (e.g., drop, rate-limit) are applied. Otherwise, the packet proceeds to the next phase, where flow-level features are extracted and evaluated by a local inference model.

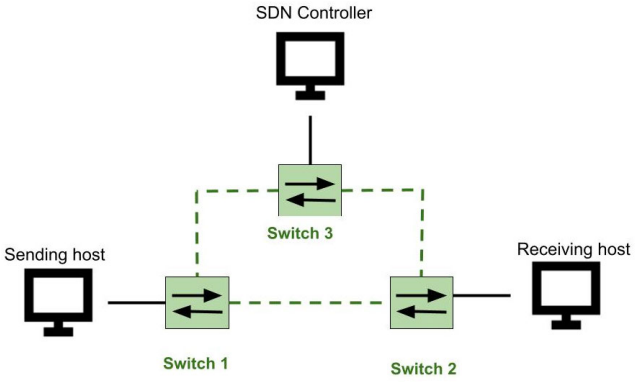

FIGURE 4. A simplistic topology in Mininet for proof of concept.

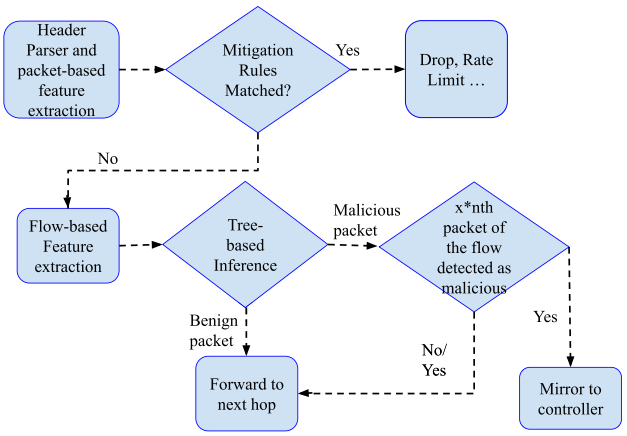

FIGURE 5. Working of the switch dedicated for inference of malicious flows.

```
hdr.ipv4_option.spkts = (bit<16>)(meta.spkts);
hdr.ipv4_option.dpkts = (bit<16>)(meta.dpkts);
hdr.ipv4_option.attack_packet= (bit<16>)(meta.attack_packet);
```

FIGURE 6. A code snippet in P4 detector script, sending telemetry information in an IPOption header of a *malicious* packet to the controller.

If a packet is classified as benign, it is forwarded to the receiving host via the port connected to Switch 2. If identified as malicious, it is mirrored to the SDN controller through Switch 3 for mitigation, while the switch continues forwarding the flow until it either completes or is terminated by updated rules received from the controller.

Switches 2 and 3 function solely as forwarders without additional match-action rules. To simplify and enhance robustness, every n^{th} (i.e., 3^{rd}) packet within a flow identified as *malicious* triggers transmission to the controller. The payloads of these packets are dropped, and their headers are updated before being sent to the controller. For improved attack flow detection accuracy, an IPOption header is added to these packets' headers. This header stores the count of packets recognized as *malicious* within a flow, using `attack_packet` field as well as `spkts` and `dpkts`, as is

TABLE 2. Features used to train an attack detection model in `IoTShield`.

Feature Name	Description	Importance
sbytes	Source to destination bytes in a flow till now.	0.0358
flow_bytes	Total bytes transmitted in either direction in a flow till now.	0.0155
spkts	Number of packets transmitted by a source in a flow till now.	0.0024
dpkts	Number of packets transmitted by a destination in a flow till now.	0.0053
duration	Recent inter-packet duration in a flow in seconds.	0.003
min_duration	Minimum inter-packet duration in a flow in seconds.	0.0539
max_duration	Maximum inter-packet duration in a flow in seconds.	0.0359
total_duration	Total time duration of a flow till a packet, in seconds.	0.0032
mean_duration	Average inter-packet duration of a flow, in seconds.	0.0127
srate	Rate of packets transmitted in flow from source to destination in a second.	0.020
drate	Rate of packets transmitted in flow from destination to source in a second.	0.0056
rate	Sum of packets transmitted in flow from destination to source in a second.	0.738
N_IN_Conn_Src_IP	Number of out-going connect from the source IP address.	0.0357
N_IN_Conn_Dst_IP	Number of incoming connections to the destination IP address.	0.0108
avg_packet_len	Mean packet length of spkts in a flow.	0.0031
state_int	If this is first packet in a UDP flow.	5.6×10^{-5}
state_rst	If this is a TCP packet and reset flag is set.	9.85×10^{-5}
ttl	Time to live field value in an IP header.	0.0169

shown in Figure 6. Total number of received packets for the flow are computed by suming up `spkts` and `dpkts`.

The CNN attributor, in the SDN controller, runs when it receives multiple triggers of malicious packets. The attributor takes into account only the most recent packet received from a flow. It computes the ratio of `attack_packet` and the total number of packets, received in the flow by the P4 switch up to that point. The ratio is rounded to the nearest integer and if it equals 1, the flow is considered *malicious* and a candidate for attribution. Subsequently, after attribution, the switch rules are updated to mitigate similar attack flows in the future.

V. EVALUATION

A. DETECTION OF MALICIOUS FLOWS

Increasing the depth and number of trees in a RF model typically enhances its accuracy. To explore this, we assessed three RF models, all with a depth of 10. These models consisted of one, three, and five trees, respectively. The Figure 7 (a) presents the percentage of detected flows for each model type. Despite variations across different flows, the maximum accuracy improvement observed was only up to 2% when transitioning from a single-tree model to its corresponding model with five trees. Given this insignificant accuracy enhancement, we opted to maintain the less complex RF model with a single tree i.e. a DT.

Furthermore, we examined the influence of depth while keeping the number of trees constant at one. We tested RF models with depths of 5, 10, 15, and 20. In Figure 7 (b), it is illustrated that false flags are minimal at a depth of 10, while the detection of *malicious* activity improves with increasing depth. To strike a suitable balance between distinguishing *benign* and *malicious* flows and maintaining lower complexity, we settled on a tree depth of 10. At this depth, the accuracy of detecting *malicious* flows is only 2% lower than the best available results.

To evaluate the accuracy of the DT model on the P4 switch, we replay PCAP files of 1MB each, for benign traffic, as well as the four attack types listed in Table 1. Using a single tree (i.e., a DT) and a depth of 10, we assess the performance of the detector with the metrics presented in Table 3. We achieve *Detection Rates* of 94%, 80%, 99.9%, and 98.8% for TCP Flood, ACK Fragmentation, RSTFIN, and SYN floods respectively, in terms of identifying malicious flows. However, the *False Positive Rate* remains around 2%.

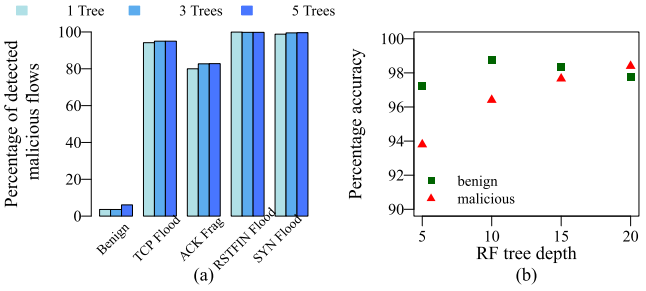

FIGURE 7. (a) Percentage of flows detected as *malicious* with different number of RF trees (b) The accuracy of RF model with single tree under different depths.

B. ATTRIBUTION OF FLOWS MARKED AS MALICIOUS

To reduce bandwidth overhead from switch to controller, every 3^{rd} packet in a flow, detected as an *malicious*, is mirrored towards SDN controller via switch 3. To enhance certainty, of whether the received packet depicts a *malicious* flow, only the latest packet from a flow is used in attribution process. The telemetry information in IPOption header

TABLE 3. Performance metrics.

Metric	Equation
Detection Rate or Recall	$True\ Positive/(True\ Positive + False\ Negative)$
False Positive Rate	$False\ Positive/(False\ Positive + True\ Negative)$
Precision	$True\ Positive/(True\ Positive + False\ Positive)$
F1 score	$2 \times (Recall \times Precision)/(Recall + Precision)$

TABLE 4. Results for DDoS attacks classification.

Attack Type	Precision	Recall	F1 score
ACK-Fragmentation	0.98	0.85	0.91
RSTFIN Flood	0.95	1.00	0.97
TCP Flood	1.00	1.00	1.00
SYN Flood	1.00	1.00	1.00

(see Figure 6), of the transmitted packet, is then extracted to compute the fraction of packets in the flow, that were triggered as anomalous by the P4 switch, if the ratio rounds to 1, the flow is assumed to be an *malicious* flow and it is attributed, otherwise ignored.

For the attribution, `IoTShield` efficiently classifies different DDoS attacks within the controller, achieving an accuracy of 99% for all attacks. Specifically, the CNN model excels in identifying TCP Flood and SYN Flood attacks, maintaining an impressive F1 score of 100%. Its performance slightly declines to F1 score of 97% and 91% when detecting RSTFIN Flood and ACK-Fragmentation attacks, respectively.

C. IMPACT ON NETWORK PERFORMANCE

To measure the time a packet spends on a switch that detects malicious flows, we utilize the parameter `deq_timedelta` from the standard metadata structure[1] of the V1Model architecture. This parameter records the time, in microseconds, that the packet remains in the queue between the ingress and egress pipeline. We observe this parameter when running an attack detector model on a switch and compare it to running a basic switch that simply forwards packets between different network segments.

Based on our minimal Mininet simulator, we find that the detector switch increases the queuing delay by a median of 0.14 milliseconds (see Figure 8 (a)). According to [32], end-to-end delay starts to become noticeable for highly demanding interactive applications when it exceeds 20 *ms*, or according to [33], when it surpasses 50 *ms*. With a median queuing delay of 0.2 *ms*, in-network attack detection can lead to a 20 *ms* end-to-end delay only when there are at least 100 switches along the end-to-end path, and each switch functions as an attack detector.

If attack detection is not performed at the switch level, an alternative approach involves utilizing the network controller or a designated entity as an IDS. In this scenario, let us assume the responsibility falls on the SDN controller. In this setup, switches will primarily focus on forwarding, while

[1]https://github.com/P4lang/behavioral-model/blob/main/docs/simple_switch.md

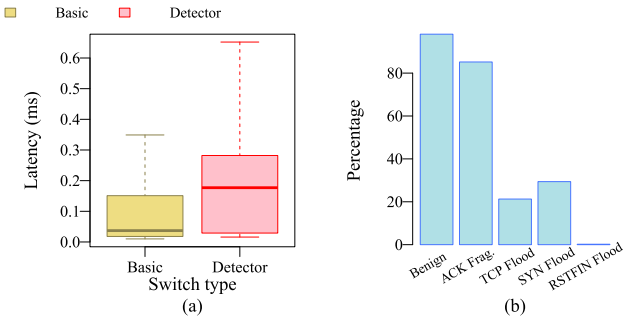

FIGURE 8. (a) Processing time in switch with and without the attack detector model and (b) Percentage drop in data transfer, to the controller, with the switch-based attack detector.

packets will be mirrored to the controller for detection and attribution.

While this approach reduces the load on switch resources and queuing delays for packets, it introduces overhead on the links between switches and the controller. By measuring link utilization in terms of transferred packets, both with and without employing a detector at the switch, we observe a potential of upto 100% decrease in link utilization with the detector. Figure 8 (b) illustrates the percentage decrease in link overhead across different types of flows. It is important to note that in this scenario, we assume the detector mirrors every $n^{th} = 3^{rd}$ detected malicious packet of a flow to the controller.

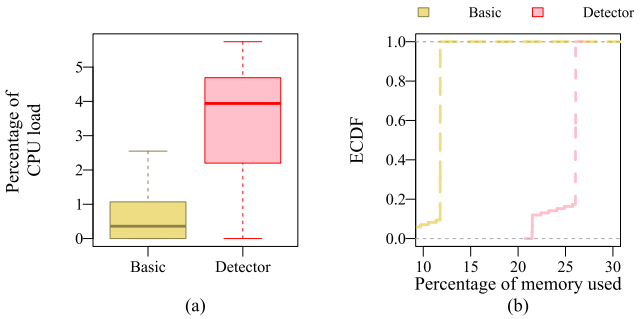

FIGURE 9. Resource utilization on a switch, with and without the attack detector model.

Finally, we evaluate the resource consumption of a detector on a P4 switch. To measure this, we employed the System Activity Report (SAR)[2] tool to monitor CPU load and memory usage. As illustrated in Figure 9, the CPU load increases from 0.5% to 4% on average when executing an attack detector on a P4 switch. The memory utilization on the detector escalates from 12% to 27%. This increase in load is attributed to the utilization of multiple stateful elements, particularly `registers`, within the P4 script.

VI. DISCUSSION AND LIMITATIONS

One might question whether employing a classification model directly at a P4 switch, rather than utilizing dual-stage

[2]https://man7.org/linux/man-pages/man1/sar.1.html

TABLE 5. Comparison of related work with `IoTShield`.

Study	Tested Malicious Flows	ML model	Inference location	Detection	Attribution	Mitigation
pForest (27)	Port scan, DDoS, botnet and brute force	RF	Data plane	✓	✗	✗
switchTree (28)	Volumetric DDoS	RF	Data plane	✓	✗	✗
Q. Qin et. al (41)	Multiple types	BNN	Data plane	✓	✗	✗
BACKORDERS (38)	Low bandwidth/ slow DDoS	RF	Data plane	✓	✗	✗
G. Siracusan et. al (43)	Port scan and DoS	BNN	Data plane	✓	✗	✗
Mousikav2 (44)	Service scanning and DDoS	BDT	Data plane	✓	✗	✗
Bungee-ML (45)	DDoS	NB, RF	Controller	✓	✗	✓
IoTShield	Multiple DDoS and Web-based attacks	DT, CNN	Data plane and Controller	✓	✓	✓

attack detection and classification, could be more efficient. Integrating benign flows into the classification process on a switch, however, often leads to significant inaccuracies. In our experimentation, training a classifier on five distinct classes, encompassing benign flows and the four types of DDoS attacks, revealed significantly degraded results. Notably, the classification accuracy plummeted to nearly 0% for two specific attack types: TCP Flood and ACK-Fragmentation. Meanwhile, the false alarms for benign flows surged to 33%.

While the dual-stage defensive framework of `IoTShield` exhibits commendable performance in our experiments, the study is not without its limitations:

- Initially, `IoTShield` was tested on a software-based prototype, implementing the RF model in the P4-data plane using the V1Model architecture. As part of our future endeavors, we aim to assess `IoTShield` on hardware programmable switches.
- Secondly, various attack categories display distinct patterns, necessitating separate training for their detection. Additionally, differences may exist in the optimal predictor features for each category. For instance, in subsection III-C, although the false flag rate of the Web-based attack detector was 11%, considerably lower than the combined DDoS and Web-based attack detector, it remains relatively high. We attribute this to the reuse of the feature set in Table 2 as predictors, without thorough exploration for the Web-based attack detector. In forthcoming research, we intend to explore the creation of detectors for different prominent attack categories, employing suitable predictor sets.
- Thirdly, as shown in subsection III-C, the false negatives for the web-based attack detector are higher than those of the combined model. This suggests that certain features of web-based attacks may closely resemble those of benign traffic, leading to misclassification. We hypothesize that even attack types within the same category can exhibit varying flow patterns. Therefore, a more in-depth analysis is necessary before deciding whether to train a single detector or multiple specialized detectors for different attack types.
- Lastly, while this paper focuses on a tree-based model, we acknowledge the potential of mapping different machine learning algorithms to a match-action pipeline, such as BNN. Furthermore, efforts are required to reduce the load and packet queuing delay at the switches.

It is important to note that all these limitations and considerations serve as focal points for our ongoing and future work.

VII. RELATED WORK

There have been numerous applications of programmable data planes, including congestion detection [34], in-network telemetry [35], load balancing [36] and traffic engineering [37], to name a few. In addition to these, there is a growing trend in proposals for machine learning methods to detect malicious activity within the data plane. In a recent study, B. Coelho and A. S-Filho [38] introduced BACKORDERS, which employs RF to identify DDoS attacks in a programmable switch using the CICIDS2017 dataset [39]. Similarly, in [27], the authors presented pForest, an in-network RF model for attack detection, utilizing both the CICIDS2017 and UNIBS-2009[3] datasets. Furthermore, in [28], J-H Lee and K. Singh proposed SwitchTree, which also utilizes RF to detect DDoS attacks, this time using the UNSW-NB15 dataset [40].

Q. Qin et al. [41] proposed the use of BNN and federated learning for Intrusion Detection (ID), utilizing the CICIDS2017 and ISCX Botnet 2014 [42] datasets. Similarly, G. Siracusano et. al [43] suggested employing BNN in the SmartNIC of mainstream systems in data centers. Instead of conducting feature extraction at the NIC data plane and relaying traffic analysis and inference to the systems' CPU, the study applies BNN-based inference directly at the NIC. They assessed their approach for IoT traffic classification, security anomaly detection, and network tomography against DT and RF, as well as traditional CPU-based inference. Given the computational expense of the Neural Network model, G. Xie et al. [44] introduced Mousikav2. In Mousikav2, the authors deploy a BDT in the data plane to perform flow-type

[3]http://netweb.ing.unibs.it/ ntw/tools/traces/

classification, application-type classification, and malware detection independently. To address the weak accuracy of BDT, they employ a teacher-student knowledge distillation method, where a more sophisticated ML model (including RF, GBDT, GRU, LSTM, and MLP) is trained on the corresponding dataset outside the data plane on a high-processing machine. Despite the aforementioned studies proposing the detection of malicious flows, mostly focusing on a single type of DDoS attack, none address the classification and mitigation of various types of attacks that may affect a network system.

The work closest to ours is Bungee-ML [45], which leverages the fast processing speed of the data plane and the high capacity and intelligence of the control plane to mitigate DDoS attacks. Bungee-ML continuously monitors traffic at the data plane to detect traffic anomalies and provides machine learning models (running in the control plane) with inputs to conduct in-depth traffic analysis. However, this analysis is not aimed at classifying the attack type but rather verifying whether an attack actually occurred on the network.

VIII. CONCLUSION

This paper presents `IoTShield`, a novel dual-stage framework designed to protect IoT networks against a wide range of malicious flows at line-rate performance. Built on an SDN-based architecture, `IoTShield` leverages machine learning-based detectors integrated into the programmable data plane to identify prevalent attacks, while the control plane oversees network management and security orchestration. With global network visibility, the controller intelligently distributes detection responsibilities across switches, ensuring malicious traffic is intercepted as close to its origin as possible.

`IoTShield` utilizes a lightweight Decision Tree model for real-time flow classification, translating its logic into match-action rules compatible with P4-enabled switches. In parallel, a CNN -based classifier in the control plane performs fine-grained attack attribution, enabling dynamic updates to switch rules for blocking identified threats. Experimental results demonstrate that `IoTShield` achieves high detection accuracy while maintaining line-rate processing. Moreover, it significantly reduces the burden of centralized detection on the controller, enhancing the scalability and resilience of IoT network security.

REFERENCES

[1] L. S. Vailshery, "Number of IoT connected devices worldwide 2019-2021, with forecasts to 2030," *Retrieved September*, vol. 8, p. 2021, Sep. 2022.
[2] R. Vishwakarma and A. K. Jain, "A survey of DDoS attacking techniques and defence mechanisms in the IoT network," *Telecommun. Syst.*, vol. 73, no. 1, pp. 3–25, Jan. 2020.
[3] G. Acar, D. Y. Huang, F. Li, A. Narayanan, and N. Feamster, "Web-based attacks to discover and control local IoT devices," in *Proc. Workshop IoT Secur. Privacy*, Aug. 2018, pp. 29–35.
[4] C. Faircloth, G. Hartzell, N. Callahan, and S. Bhunia, "A study on brute force attack on T-mobile leading to SIM-hijacking and identity-theft," in *Proc. IEEE World AI IoT Congr. (AIIoT)*, Jun. 2022, pp. 501–507.
[5] S. Badotra, D. Nagpal, S. N. Panda, S. Tanwar, and S. Bajaj, "IoT-enabled healthcare network with SDN," in *Proc. 8th Int. Conf. Rel., INFOCOM Technol. Optim. (Trends Future Directions) (ICRITO)*, Jun. 2020, pp. 38–42.
[6] A. Dutt, V. P. Singh, H. B. Pasupuleti, and S. SD, "Adaptation of SDN framework for a smart water distribution network in smart cities," in *Proc. 3rd Int. Conf. Mobile Netw. Wireless Commun. (ICMNWC)*, Dec. 2023, pp. 1–6.
[7] A. Petrosyan. (2023). *Global Annual Number of Iot Cyber Attacks 2018–2022*. Accessed: Mar. 2, 2024. [Online]. Available: https://www.statista.com/
[8] T. Lacoma. (Apr. 2025). *New Reports Say Smart Device Cyberattacks More Than Doubled in 2024: Should You Worry*. Accessed: Apr. 11, 2025. [Online]. Available: https://www.cnet.com/home/security/smart-device-cyberattacks-more-than-doubled-in-2024-should-you-worry/
[9] O. Abimbola and O. O. Idris, "A critical cybersecurity analysis and future research directions for the Internet of Things: A comprehensive review," *Path Sci.*, vol. 11, no. 3, p. 4009, Mar. 2025.
[10] M. Zang, C. Zheng, L. Dittmann, and N. Zilberman, "Toward continuous threat defense: In-network traffic analysis for IoT gateways," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 9244–9257, Mar. 2024.
[11] M. A. Ja'afreh, H. Adhami, A. E. Alchalabi, M. Hoda, and A. El Saddik, "Toward integrating software defined networks with the Internet of Things: A review," *Cluster Comput.*, vol. 25, no. 3, pp. 1–18, Jun. 2022.
[12] A. Abderrahmane, H. Drid, and A. Behaz, "A survey of controller placement problem in SDN-IoT network," *Int. J. Networked Distrib. Comput.*, vol. 12, no. 2, pp. 1–15, Dec. 2024.
[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.
[14] M. Saqib, H. Elbiaze, R. H. Glitho, and Y. Ghamri-Doudane, "An intelligent and programmable data plane for QoS-aware packet processing," *IEEE Trans. Mach. Learn. Commun. Netw.*, vol. 2, pp. 1540–1557, 2024.
[15] D. Sanvito, "Traffic management in networks with programmable data planes," in *Special Topics in Information Technology*, Feb. 2021, ch. 2, pp. 13–23.
[16] G. Grigoryan, Y. Liu, and M. Kwon, "ILoad: In-network load balancing with programmable data plane," in *Proc. 15th Int. Conf. Emerg. Netw. EXperiments Technol.*, Dec. 2019, pp. 17–19.
[17] Z. Zhao, F. Liu, Z. D. Meng, Q. Zhao, and X. Xie, "DTS: Dynamic traffic scrubbing against link flooding attacks with programmable data plane," *Proc. SPIE*, vol. 13447, pp. 507–513, Jan. 2025.
[18] G. Liu, W. Quan, N. Cheng, N. Lu, H. Zhang, and X. Shen, "P4NIS: Improving network immunity against eavesdropping with programmable data planes," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 91–96.
[19] C.-M. Iurian, D. Zinca, I.-A. Ivanciu, T.-M. Blaga, and V. Dobrota, "A syn flooding ddos attack detection in p4-based programmable networks," *Acta Technica Napocensis*, vol. 62, no. 2, pp. 19–24, 2022.
[20] Z.-Y. Shen, M.-W. Su, Y.-Z. Cai, and M.-H. Tasi, "Mitigating SYN flooding and UDP flooding in P4-based SDN," in *Proc. 22nd Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Sep. 2021, pp. 374–377.
[21] X. Chen, H. Liu, D. Zhang, Q. Huang, H. Zhou, C. Wu, and Q. Yang, "Empowering DDoS attack mitigation with programmable switches," *IEEE Netw.*, vol. 37, no. 3, pp. 112–117, May 2023.
[22] M. Zhang, G. Li, S. Wang, C. Liu, A. Chen, H. Hu, G. Gu, Q. Li, M. Xu, and J. Wu, "Poseidon: Mitigating volumetric DDoS attacks with programmable switches," in *Proc. 27th Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2020.
[23] Z. Xiong and N. Zilberman, "Do switches dream of machine learning: Toward in-network classification," in *Proc. 18th ACM Workshop Hot Topics Netw.*, Nov. 2019, pp. 25–33.
[24] S. Rathee, D. Uttamchandani, K. Haribabu, and A. Bhatia, "An efficient method to collect statistics in SDN using curvature based sampling," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2021, pp. 175–180.
[25] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Comput.*, vol. 24, no. 3, pp. 1999–2012, Feb. 2020.
[26] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," *Sensors*, vol. 23, no. 13, p. 5941, Jun. 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/13/5941

[27] C. Busse-Grawitz, R. Meier, A. Dietmüller, T. Bühler, and L. Vanbever, "PForest: In-network inference with random forests," 2019, *arXiv:1909.05680*.
[28] J.-H. Lee and K. Singh, "SwitchTree: In-network computing and traffic analyses with random forests," *Neural Comput. Appl.*, pp. 1–12, Nov. 2020.
[29] K. Tools. *HPING3 Package Description*. Accessed: Jan. 27, 2024. [Online]. Available: https://www.kali.org/tools/hping3
[30] (2017). *Dvwa. DAMN Vulnerable Web Application*. Accessed: Mar. 5, 2024. [Online]. Available: https://github.com/digininja/DVWA
[31] T. H. News. (2023). *Ddos 2.0: IoT Sparks New DDoS Alert*. Accessed: Jan. 1, 2024. [Online]. Available: https://thehackernews.com/2023/09/ddos-20-iot-sparks-new-ddos-alert.html
[32] K. Raaen and T. M. Grønli, "Latency thresholds for usability in games: A survey," in *Proc. Norsk IKT-Konferanse Forskning oG utdanning (Norwegian ICT Conf. Res. Educ.)*, 2014.
[33] R. E. Bailey, J. J. Arthur, and S. P. Williams, "Latency requirements for head-worn display S/EVS applications," *Proc. SPIE*, vol. 5424, pp. 98–109, Aug. 2004.
[34] M.-R. Fida, A. H. Ahmed, T. Dreibholz, A. F. Ocampo, A. Elmokashfi, and F. I. Michelinakis, "Bottleneck identification in cloudified mobile networks based on distributed telemetry," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5660–5676, May 2024.
[35] D. Scano, A. Giorgetti, F. Paolucci, A. Sgambelluri, J. Chammanara, J. Rothman, M. Al-Bado, E. Marx, S. Ahearne, and F. Cugini, "Enabling P4 network telemetry in edge micro data centers with kubernetes orchestration," *IEEE Access*, vol. 11, pp. 22637–22653, 2023.
[36] J. Zhang, S. Wen, J. Zhang, H. Chai, T. Pan, T. Huang, L. Zhang, Y. Liu, and F. R. Yu, "Fast switch-based load balancer considering application server states," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1391–1404, Jun. 2020.
[37] S. Kamamura, "Dynamic traffic engineering considering service grade in integrated service network," *IEEE Access*, vol. 10, pp. 79021–79028, 2022.
[38] B. Coelho and A. Schaeffer-Filho, "BACKORDERS: Using random forests to detect DDoS attacks in programmable data planes," in *Proc. 5th Int. Workshop P4 Eur.*, Dec. 2022, pp. 1–7.
[39] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.
[40] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[41] Q. Qin, K. Poularakis, K. K. Leung, and L. Tassiulas, "Line-speed and scalable intrusion detection at the network edge via federated learning," in *Proc. IFIP Netw. Conf. (Netw.)*, Jun. 2020, pp. 352–360.
[42] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *Proc. IEEE Conf. Commun. Netw. Secur.*, Oct. 2014, pp. 247–255.
[43] G. Siracusano, S. Galea, D. Sanvito, M. Malekzadeh, G. Antichi, P. Costa, H. Haddadi, and R. Bifulco, "Re-architecting traffic analysis with neural network interface cards," in *Proc. 19th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, 2022, pp. 513–533.
[44] G. Xie, Q. Li, G. Duan, J. Lin, Y. Dong, Y. Jiang, D. Zhao, and Y. Yang, "Empowering in-network classification in programmable switches by binary decision tree and knowledge distillation," *IEEE/ACM Trans. Netw.*, vol. 32, no. 1, pp. 382–395, Feb. 2024.
[45] L. A. Q. González, L. Castanheira, J. A. Marques, A. E. Schaeffer-Filho, and L. P. Gaspary, "Bungee-ML: A cross-plane approach for a collaborative defense against DDoS attacks," *J. Netw. Syst. Manage.*, vol. 31, no. 4, p. 77, Oct. 2023.

MAH-RUKH FIDA received the Ph.D. degree in measurements and analytics in mobile broadband (MBB) computing from The University of Edinburgh, U.K., and the Postdoctoral degree in performance modeling in MBB networks from Simulamet, Oslo, Norway. She is currently a Senior Lecturer with the School of Business, Computing, and Social Sciences, University of Gloucestershire, U.K. In addition to her existing work on telemetry and performance evaluation within programmable networks, her research interests include applied data analytics and AI-based modeling, particularly in the fields of the Internet of Things and secure communications.

AZZA H. AHMED received the master's degree from the University of Nottingham, in 2012, and the Ph.D. degree from Oslomet University, Norway, in 2023. She is currently a Postdoctoral Research Fellow with Simula Metropolitan Centre for Digital Engineering in Oslo, Norway. She was a Lecturer with the University of Khartoum, Sudan. Her work revolves around the measurement and analysis of mobile broadband networks performance, network performance optimization, network automation, and machine learning to solve networking problems. Her research interests include network management and control, seeking innovative solutions to enhance the operational efficiency, and reliability of these complex systems.

AMEER SHAKAYB ARSALAAN (Member, IEEE) received the Ph.D. degree from the School of Computer and Mathematical Sciences, University of Adelaide, Australia, in 2024, jointly funded by HEC Pakistan and University of Adelaide. He is currently a Lecturer with the Department of Computer Science, University of Swabi, Pakistan. He has published extensively and collaborated with industry and academic partners to bridge research with real-world applications in wireless communication. He is specializing in wireless networks, particularly MANETs, IoT, and QoI-based data transmissions, his work focuses on advanced routing protocols and adaptive algorithms to optimize network performance and fulfill user's requirements in decentralized systems. Recognized for his contributions to emergency response and autonomous vehicle networks.