IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 22, 2020, accepted December 3, 2020, date of publication December 7, 2020,
date of current version December 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3043082

An Efficient Counter-Based DDoS Attack Detection Framework Leveraging Software Defined IoT (SD-IoT)

JALAL BHAYO[1], SUFIAN HAMEED[1], AND SYED ATTIQUE SHAH[2]
[1]Department of Computer Science, National University of Computer and Emerging Sciences (NUCES), Karachi 75160, Pakistan
[2]Data Systems Group, Delta Centre, Institute of Computer Science, University of Tartu, 51009 Tartu, Estonia

Corresponding author: Jalal Bhayo (jalal.bhayo@nu.edu.pk)

This work was supported by the European Social Fund via IT Academy Programme.

ABSTRACT The Internet of things (IoT) introduces emerging applications (i.e., smart homes, smart cities, smart health, and smart gird) that assist the traditional infrastructure environments to be connected with smart objects. Things are connected with the Internet and numerous new IoT devices are developing at a rapid pace. As these smart objects are connected and able to communicate with each other in unprotected environments; therefore, the whole communication ecosystem requires security solutions at different levels. IoT technology possesses unique characteristics with various resource constraints and heterogeneous network protocol requirements, unlike traditional networks. The attacker exploits numerous security vulnerabilities of an IoT infrastructure, to generate a DDoS attack. The increase in DDoS attacks has made it important to address the consequences which imply in the IoT industry. This research proposes an SD-IoT based framework that provides security services to the IoT network. We developed a C-DAD (Counter-based DDoS Attack Detection) application that is based on counter values of different network parameters, which helps to detect DDoS attack successfully. C-DAD is a dynamic and programmable solution, and is deeply tested with different network parameters. The algorithm demonstrates a good performance with better results through SDN. Moreover, the proposed framework detects the attack efficiently in a minimum amount of time and with lesser consumption of CPU and memory resources.

INDEX TERMS SD-IoT, SDN, attack detection, DDoS, counter-based DDoS detection.

I. INTRODUCTION

IoT technology has exponentially increased the number of heterogeneous devices linked with the Internet. One of the major challenges is to provide security to these devices, which operates with low power, limited resources, and diverse access protocol. According to [1], as of usage, the number of IoT devices will reach a quantity of 24 billion by the end of this year, and IoT will exceed to 100 billion connected devices by the end of 2025 [2]. These devices spread in an open network environment, which is easy to attract any attacker. The integration of these devices with programmable and flexible networks can help detecting any type of intrusion in the IoT network.

IoT provide an infrastructure in which physical objects i.e., RFID, sensors, actuators, and other smart devices can be connected with the Internet. IoT devices can be utilized in different areas such as smart home, smart traffic system, smart healthcare services and smart industrial manufacturing to name a few. According to a research [3], 200 million IoT devices connect to the Internet and are associated with the web, so it offers an opportunity for the attackers to utilize these devices for DoS, DDoS, Trojan, and e-mail pernicious. Another issue with IoT devices is lack of computation and communication, as compared to traditional computing devices. Therefore, these devices can be compromised easily and turned into a botnet to launch a DDoS attack [4]. For example, in the 2016 attack on DNS service provider company DYN through Bashlite and Mari malware installed on different IoT devices, therefore most high profile websites services became unavailable such as Twitter, Netflix, GitHub,

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

and other many more. In the same year, there was also another attack via IoT devices through Botnet [5], in which the attacker targeted Krebs and the French WebHost using six hundred thousand compromised IoT devices with 620 Gbps and 1.1 Tbps traffic.

With evolution of IoT, millions of devices are connected to exchange information, required for various applications that are important for human lives. Meanwhile, the security issues are as vital as IoT devices exponential growth increase [6]. SD-IoT provides flexible control on network traffic generated via IoT devices by the attacker. To tackle these challenges, many researchers started to work on security issues in SD-IoT network to overcome its drawbacks. For example, Silva *et al.* [7] study carry out in-depth investigation and provides a comprehensive view of security challenges, and demonstrates main benefits and drawback of each SDN based security approaches in IoT scenarios. The study presents the IoT applications, IoT architecture and highlights the relevant issues and challenges.

The DDoS attack is very harmful which exploit normal operations of the network and consumes its bandwidth or resources. In [8], the authors describe the latest DDoS attack detection and mitigation techniques for SD-IoT network. There are many DDoS detection approaches including entropy-based statistical methods [9], machine learning-based classification of malicious traffic techniques [10], and rule-based approach [11] which are implemented using SDN paradigm.

The SD-IoT based proposed framework is divided into three-layer same as the SDN-based IoT architecture layer model, however, our model is further customized with additional components according to the problem's requirement. The first layer consists of security applications that detect the DDoS attack. It has a C-DAD attack detection application that is part of the application layer. The second layer is the control layer, which has two components, SDNWISE controller and IoT controller, to manage the SD-IoT network. The third layer is the infrastructure layer, which is also named as a data plane that contains SOFS (Sensor Open Flow Switch) and IoT devices.

The application layer's first module consists of C-DAD that detects the DDoS attack in the SD-IoT network. The architecture of the C-DAD depends on different sub-module, which consist of counters. C-DAD evaluate the counter value of different sub-modules counters, and decide whether traffic is malicious or legitimate. Each counter has a threshold value. If it exceeds that value, it generates a trigger to flow analyzer, which checks all counters' status and decides whether the network is compromised or normal. There is a separate subsection for this module in which different experiments with different variables and associated algorithms are discussed in detail. Each experiment is further deeply analyzed for every variable with different parameters meanwhile each experiment with set variable and associated algorithms and results are further showed in tabular and graph format so it easy to understand.

In the conventional systems, the Intrusion Detection System (IDS) security instrument is implemented at the edge level of the Internet. These solutions are utilized to keep the system safe from external threats. Such systems are not sufficient to handle the security of the cutting-edge Internet. IoT is based on the border-less system architecture, which raises an extra threat to the system access control. However, security is a major issue for an ad-hoc system in IoT.

This research work aims to contribute towards a software defined-IoT security platform that provides security services to its network. The three main contributions of this work are as followings,

- SDNWISE based customized framework for IoTs network to provide the security services. This framework has a dynamic feature to add security services at the top of SDNWISE with minimal programming and configuration without changing the network infrastructure. The framework consists of an IoT controller, which is used as a gateway between the IoT network and SDNWISE controller to connect different heterogeneous IoT networks and SOPFS (Sensor Open Flow Switch), which provides the communication between IoT Nodes and path toward the SDNWISE controller via IoT controller.
- Counter-based DDoS Attack Detection (C-DAD) application, which is running on top of the proposed framework that consists of different counter-based methods like Packet Counter, Payload Counter, Traffic workload counter, and others to determine the malicious traffic and to detect DDoS attack in SD-IoT Network. This algorithm is most efficient for zero-day attacks due to its functionality based on counter's values and because it not based on an attack signature or machine learning-based trained model.
- According to simulation results, the C-DAD algorithm easily detects the DDoS attack in the SD-IoT network by analyzing different parameters with three different types of experiments. The algorithm is deeply tested and analyzed with each parameter separately to get the best outcomes. The result shows that it detects DDoS attacks very efficiently.

The rest of this article is as follows; Section-II discusses the background for the research work. Section-III investigates the related work. Section-IV presents the proposed SD-IoT framework and its components, and Section-V discusses the experiment evaluation and results along with discussion. Finally, Section-VI concludes the research work.

II. BACKGROUND

A. SOFTWARE DEFINED IoT ARCHITECTURE

The IoT networks are facing various challenges due to heterogeneous nature from access technology, routing protocols, geographically distributed and heterogeneous network infrastructures. Also, various solutions with different approaches are found in literature for SDN based IoT architecture, but no standard is followed [12]–[16]. A programmable network

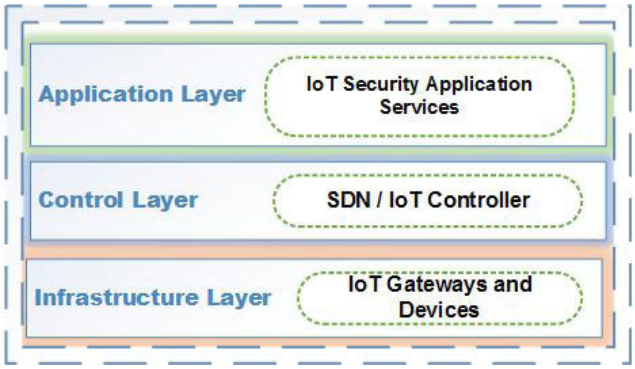


FIGURE 1. SDN-based IoT architecture.

is initiated by many researchers to develop novel SDN paradigms [17], [18], that can be useful to develop a layer based architecture for IoT as shown in Figure. 1.

The architecture consists of three layers i.e., Application layer, Control layer, and Infrastructure layer. IoT network, sensors, and user devices are included controlled through the control plane in the Infrastructure layer. In the SD-IoT network, infrastructure layer's devices act as a forwarding device to forward traffic according to network flow. The network is divided into two parts: control plan and data planes. In the control plane, it includes IoT devices and openflow switches, but the control plan consists of an IoT controller and SDN controller to perform network operations such as routing, defining network rules, policy etc. The control layer provides network control services such as access control, authentication, integrity, network and system security, Big-data security analysis, and confidentiality. The application layer consists of IoT security services, which users and administrators use to control the SD-IoT environment.

SDN-based IoT architecture [19] proposal is interoperable, scalable and adaptable. It must connect wireless network IPv6 over Low-Power Wireless Personal Area Networks (6LowPAN) and Routing Protocol for Low-Power and Lossy Networks (RPL) and communicate with the gateway for fetching and sending the data for web services. Security and QoS parameters are also negotiated during the connection establishment process based on M2M tweaked architecture. CoAP protocol is implemented to Fateh and sends the data to the IoT node using the IEEE 802.15.4 standard for communication. A secure IoT architecture is presented in [20], which is based on four architectural blocks, including trusted SDN controller, black network, unified registry, and key management for the smart city. These four components provide the security to the smart city like privacy, identity management and authentication, secure routing, and secure key management.

B. SDNWISE IoT NETWORK ARCHITECTURE

SDNWISE provides the software framework based on Software-defined Wireless Sensor Network (SD-WSN) and hardware prototype. SDNWISE has two main features; first, through SD features, it reduces the information exchanged between node and controller compared to non-SDN; second, with the help of SDN, sensor nodes become programmable [21].

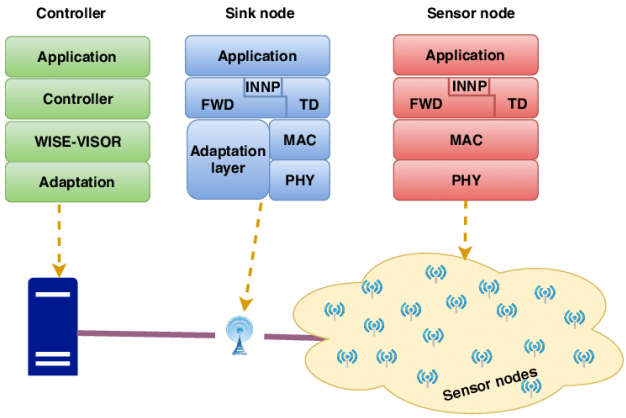


FIGURE 2. SDNWISE IoT architecture [22].

Figure. 2 shows the SDNWISE IoT architecture. It can be divided into two main parts; the control plane and data plane. The sensor node and sink node are part of the data plane, which in most architectures is provided with the same functionalities, but here few differences are discussed below. The controller is the part of the control plane [22], and it has different protocol stack layers which are described in the subsequent section.

The SDNWISE defines the open-source controller to perform the routing decision based on the Dijkstra algorithm among the IoT nodes. These IoT nodes collaborate with controller via sink node or IoT network gateway/IoT controller. The WSN is based on the SDNWISE tool, which supports 6LowPAN protocol for low power devices, so sensors can communicate with IPv6 devices that are used in IoT network environments such as smart home, smart city, etc. [23].

1) CONTROL PLANE

The control-plane's sub layers are listed as followings,

- **Application.** The network applications are running on the top SDNWISE controller and use its API to provide the network services.
- **Controller.** The control layer builds the control logic. It also defines the network policies that are implemented by IoT nodes.
- **WISE-Visor.** It contains the topology management layer, which is used to provide the abstraction of network resources.
- **Adaptation.** The Adaptation layer plays a mediator between the controller and the IoT network. Its major functionalities include formatting messages such as 6LowPAN into IEEE 802.15.4 and vise-versa to support IPv6 communication to understand both the sensor and controller easily.

2) DATA PLANE

The data-plane's sub layers are listed as followings,

- **Application.** The applications are running on IoT nodes that provide services to the IoT node in the SD-IoT network.
- **Topology Discovery (TD).** TD can gather the local information of nodes in the network and controls their behavior.
- **In-Network Packet Processing (INPP).** INPP is responsible for data aggregation or in-network processing operations.
- **Forwarding (FWD).** The forwarding layer includes an IEEE 802.15.4 transceiver and a micro-control unit (MCU), which manages all incoming packets.
- **Adaptation Layer.** An adaptation layer accommodates IPv6 packets into IEEE 802.15.4 frames. 6LoWPAN is mostly leveraged in embedded devices which are used in home and building automation or health-care automation [23].
- **MAC.** This Media Access Control layer supports IEEE 802.15.4 [24] standard to enable IPV6 6LowPAN communication with the IEEE help 802.15.4 frame. It is designed for low power, low-cost implementation, and low complexity features to support short-range communication for the application, which requires low power transmission.
- **PHY.** This layer supports band 868/915 MHz and 2.4 GHz low power for short-range communication. The sensor nodes operate with low power. To maximize their residual power is one of the significant challenges of this layer.

III. RELATED WORK AND EXISTING FRAMEWORKS

SDN based IoT security framework proposes security against DoS attacks in IoT networks. The IoT network is divided into three different segments; each segment has sensor nodes, mobile nodes, and smart objects. Segments fulfill the SDN requirement with OpenFlow capable and controller. SDN controller controls all traffic and also authenticates the devices within the segments. All controllers of a particular segment exchange their security rules so that only authorized nodes can communicate within the segment and even with other segments. Each node sends the request to the local controller, which forwards the destination gateway controller's request to check the requested node's flow. If it knows the node's destination address, then two different segments' endpoint devices communicate with the other [25].

The study [26] has discussed different issues and challenges in SDN based IoT network. Security is one of the significant problems in the IoT network, due to its spreading nature in the open space. Statistics regarding the network are logged, and then different applications are activated to detect the suspected malicious flows. SDN provides the flexibility to develop the application, which has been used to inspect the traffic, detect anomalies in the IoT network, and handle the DDoS attack. Software-Defined Internet of Things based framework can be used for DDoS attack detection and mitigation in IoT network with heterogeneous vulnerable devices [27]. The framework consists of a pool of controller, SD-IoT switches with integrated IoT gateway, and IoT devices. The DDoS attack detection and mitigation algorithm use cosine similarity of the vectors of packet-in rate and the threshold values to detect and mitigate the attack. According to their results, the algorithm quickly finds the attacking device in the IoT network.

Bull *et al.* [28] propose SDN gateway to monitor the traffic, which originates from and to IoT devices; the gateway detects anomalous traffic behavior and performs the appropriate response. It provides flow-based security to attack, which is based on TCP and IMCP flood. The attack detection performance of the gateway depends upon the number of flows installed per second; it can successfully detect and perform an action such as blocking, forwarding or apply QoS, which depends upon the flow. Luo *et al.* [29] develop honeypots to detect the DDoS attack in IoT. These honeypots are based on MTD (Moving Targeted Defense) architecture using SDN to mimic IoT devices and luring attackers and malware. In this approach, network resources or assets hide from a scanner via MTD and SDN-based honeypot to protect against DDoS attacks in IoT.

Ravi and Mercy Shalinie [30] presented LEDEM security solution that detects the DDoS attack in IoT network via SDN-Cloud architecture. Experiments were conducted on the testbed and emulated topology, and the result has been compared with the state-of-art solutions. LEDEM uses a supervised machine learning algorithm to detect and mitigate the DDoS attack with an improved accuracy rate of 96.28% in detecting DDoS attack. Silveira *et al.* [31] has proposed the Smart Detection-IoT module that uses Machine Learning techniques to classify the network traffic concerning SDN to IoT controller. The experiment was conducted on an emulated platform with an actual and well-known dataset. Attack (DR) detection rate and PREC were higher than 96% with False Alarm Rate less than 6%. Yang *et al.* [32] propose SDN-based IoT gateways that detect and mitigate DDoS attacks at the edge of the network. The gateway consists of three modules: Learning Module, Detection Module, and Flow Management Module. The Learning Module trains the classification algorithm of the machine learning based on collected flow's statistics of the network. The detection module classifies the malicious flows and notifies its source address for the further blocking action, which is performed by the Flow Management Module through defined filtering rules.

Flow-based SDN gateway detects the DDoS attack, which is based on a network traffic statistics analysis mechanism. Different detection methods are based on self-organizing maps, which are used to detects malicious traffic flows. Moreover, the quantity of flow can also be used in the detection method. The authors in [28], proposes an IoT gateway that acts as both an SDN switch and an integrated controller to analyses the traffic patterns with respect to its corresponding flows. After detecting the malicious flows, it executes the mitigation action against the malicious flow.

The study [33] conducted a state-of-the-art survey for techniques and methods used for Intrusion Detection Systems (IDS) proposed for the IoT. The research by Mustapha and Alghamdi [34] addresses the security issues in the form of a DDoS attack in the IoT network. IoT devices are low powered and low processing, so they are vulnerable to security issues like authentication and authorization. DDoS attacks are also classified based on their relevant solution. This article has highlighted the previous answer to prevent DDoS attacks in IoT and discusses one of the reliable solutions based on SDN and Network Functions Virtualization (NFV). The main benefits of the SDN programmable network are to provide better traffic analysis to detect a DDoS attack earlier as it happens.

The DDoS attack detects through a backtracking algorithm to find the real source, but it increases the response time [35], [36]. The DDoS attack detection, prevention, and mitigation approaches are directly deployed on IoT networks, but these strategies consume many IoT resources and might disable the IoT network during huge DDoS attacks [35]. The rule-based approaches extract the features of different DDoS and send these features at the flow table to avoid DDoS attacks. Rule-based approaches' main benefit is a high accuracy, but the disadvantage is that the features are re-extracted whenever a new type of attack is found [37].

Cui *et al.* [38] proposed an attack detection and mitigation mechanism based on inspired cognitive computing with source and destination IP address entropy values. It takes these entropy values as features vectors and uses the SVM algorithm to obtain DDoS attack detection mode. Mousavi and Mousavi [39] proposed an entropy-based DDoS attack detection algorithm, which calculates the entropy of destination IP address from incoming packets, threshold, and sampling time through SDN controller. The algorithm detects a DDoS attack if the entropy value is greater than the threshold value, but it cannot locate the specific IP address. In the machine learning approach, the machine is trained to learn malicious traffic flows and get the best learning model. According to it, the learning model can efficiently detect the DDoS attack, but it cannot detect the zero-day attack.

Yin *et al.* [27] proposed the SD-IoT framework, which detects DDoS attack based on cosine similarity of the vectors of the packet-in message rate at the boundary of SD-IoT switches. The algorithm uses threshold values of cosine similarity to decide about DDoS attack and find the DDoS attacker. But this algorithm only works on packet-in messages, and our proposed C-DAD framework used different network parameters to decide about DDoS attack.

Dehkordi *et al.* [40] proposed a DDoS attack detection mechanism based on machine learning and the statistical method. The detection method consists of the collector, entropy, and classification. The dynamic threshold with a statistical entropy-based method produces the best result with high FPR (False Positive Rate). To overcome this issue, the classification-based algorithms are run to get more accurate results. The accuracy of this method is higher than other similar methods, but the attacks are detected only by one controller, whereas, our work support to add multiple IoT controllers to connect different IoT networks.

Galeano-Brajones *et al.* [41] proposes an entropy-based DDoS detection and mitigation method using a stateful SDN data plan. The mechanism is based on Open State, in which Openflow switch is extended with monitoring capability to monitor and analyze the network information independently from the controller. The monitoring information gathered from flows and state tables at the data plan. The entropy algorithm runs on monitoring information to detect malicious flow. The result demonstrates that the main advantage of using the correlation of entropy values of different features quickly detects the DDoS attack and mitigates with the help of SDN to remove the flow entry from the switch. There is an issue in this work. Whenever the switch windows size increases and becomes large, it does not respond to the controller about the states.

Ujjan *et al.* [42] uses sFlow and adaptive polling sample approaches with the help of deep learning to detect the DDoS attack in SDN. This proposed system is based on Stacked Auto Encoders (SAE), deep neural network model using packet-based and time–based sampling. sFlow techniques overcome the network flow's windows size issue and reduce processing and network overhead of switches. The sFlow collector gathers the sampled packets and then rebuilds the new flow pattern and updates the packet counter for each flow entry. The main aim of sFlow is to reduce the flow table entry and forward the flow statistics to the detection model. The adaptive polling sample approach uses proportional linear prediction (PLP) and weighted linear prediction (WLP) to estimate the next flow. Initially, they implement the low pass filter to predict the next flow rate. This work demonstrates higher detection accuracy with 95% TPF and less than 4% FPR within sFlow based implementation as compared to adaptive polling.

Shohani and Mostafavi [43] proposed the DDoS attack detection method which overcomes the drawback of entropy and PCA method. The method detects the DDoS attack in three phases; fetching, estimation and detection. In the first phase, network statistical information fetched of receiving packets and table misses for each switch, and secondly, the dynamic threshold line equation is estimated through EWMA method, and lastly, the table misses are compared to the estimated value of the threshold to detect the DDoS attack. The model detects the DDoS attacks which are not detectable through entropy method, but this model is not suitable for SD-IoT traffic, it only detects the attack for traditional SDN based network.

Wani and Revathi [44] proposes SD-IoT- based security mechanism for DDoS detection in SD-IoT network. This research uses MCOD and MLP approaches to identify the abnormal behavior of IoT networks. It depends on SDNWISE that best to provide a stateful solution for IoT.

TABLE 1. Literature review summary with identified research gaps.

S. No	Author Year	Detection method	Platform	Deployment	Description of gaps
1	Dehkordi et al. 2020 [40]	Entropy Machine ~Learning	SDN	Controller	The attacks are detected only by one controller. Whereas, our work supports the addition of multiple IoT controllers to connect different IoT networks.
2	Brajones et.al 2020 [41]	Entropy	SD-IoT	Both	Whenever the switch windows size increases and become large, it does not respond to the controller about the states.
3	Ujjan et. al 2020 [42]	Deep Learning	SDN	Controller	This work demonstrates higher detection accuracy with 95% TPF and less than 4% FPR within sFlow based implementation as compared to adaptive polling.
4	Mostafavi et.al 2020 [43]	EWMA Linear Regression	SDN	Controller	This model is not suitable for SD-IoT traffic, it only detects the attack for traditional SDN based network
5	Wani et. al 2020 [44]	MCOD	SD-IoT	Controller	This research uses machine learning techniques to train the known features regarding malicious flow but fails to detect thee zero-day attack.
6	Cui et al. 2019 [38]	Inspired Cognitive Computing(SVM)	SDN	Controller	It takes the entropy values as features vectors and uses the SVM algorithm to obtain DDoS attack detection mode.
7	Mousavi et. al. 2015 [39]	Entropy	SDN	Controller	The algorithm detects a DDoS attack if the entropy~value is greater than the threshold value, but it cannot locate the specific IP address.
8	Yin et al. 2018 [27]	Cosine Similarity	SD-IoT	Controller	This algorithm only works on packet-in messages, but our proposed C-DAD framework used different network parameters to decide about the DDoS attack.
9	Bull et. al. 2016 [28]	Traffic Pattern Analysis	SDN-IoT Gateway	Both	Limited resources at data plan is also a major challenge to handle heavy DDoS traffic.
10	Luo et. al. 2019 [29]	SDN-Based Honeypots	SDN	Controller	This research only focused on scanning based attack. The attacker may infect the IoT network to generates huge DDoS traffic to jam the IoT network without using a scanning approach.
11	Ravi et. al. 2020 [30]	Machine Learning ELM	SDN	Controller	LEDEM uses supervised learning to detect the DDoS, the model detects the attack according to training. It also fails to detect the zero-day attacks.
12	Augusto et. al. 2020 [31]	Machine Learning	SD-IoT	Controller	This model works only for know attack's features.
13	Yang et. al. 2019 [32]	Machine Learning Edge Computing	SDN	OF Switches	The detection module classifies the malicious flows with the trained algorithm, while it was unable to detect the untrained attack.

It uses MLP and MCOD for DDoS detection. This research uses machine learning techniques to train the known features regarding malicious flow, but fails to detect the zero-day attack.

The stateful SD-IoT network has issues, whenever OpenFlow switch window size increases and becomes full, it was not able to inform the controller about the states [41]. In some approaches, only a single controller is detecting the DDoS attack, that may fail to receive a large number of unknown flow entries [40]. Our proposed system resolves these issues with the support of the multiple IoT controller. The rule-based approaches have high accuracy, but its features re-extracted, whenever they find a new attack [37].

In some approaches, detection strategies fail the IoT network due to the huge amount of DDoS traffic, hence it consumes the IoT network resources and disables the network [35]. Statistical methods based on entropy techniques that use the threshold values of the network information with uncertainty, do not have detailed information about the malicious flow. For example, with dynamic threshold value has high FPR (False Positive Rate) [40], this issue is further resolved through classification, and as another example in which entropy of destination IP address based on threshold values [39] also does not locate specific nodes.

In the last of this section, we are going to summarize different DDoS attack detection approaches with their features, gaps and also summarize where these approaches fail as shown in Table 1. Our framework is efficiently overcoming the gaps identified during this research. The proposed framework is an optimal solution because it uses different features of SD-IoT network traffic as counter values and deeply analyzes these features for DDoS attack detection.

IV. PROPOSED SD-IoT BASED FRAMEWORK

This research's main objective is to propose a framework that detects different threats, attacks, and anomalies, which are critical problems for IoT. The framework is based on SD-IoT to propose novel security solutions. The framework consists of SDN based security controller and IoT controller, which are located in an IoT gateway that communicates with IoT nodes. IoT network uses cluster-based topology, where each head node in a cluster manages a cluster of IoT devices. Figure 3 shows the proposed SD-IoT-based attack detection framework. The proposed framework is divided into layers, which is described in detail in the following subsections.

A. APPLICATION LAYER

In the SDN paradigm, this top layer uses the northbound APIs. Different applications are developed to run at the top of the controller, such as network management applications, security applications, and other network services provided to the client/organization. This research proposes two applications to detect and mitigate the DDoS attack in the SD-IoT network.

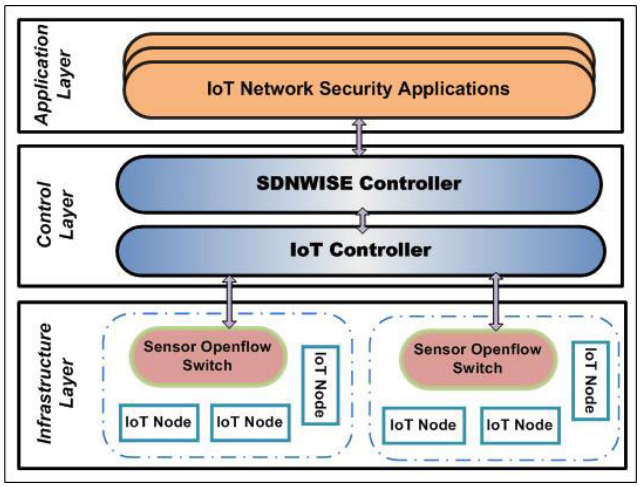


FIGURE 3. SD-IoT based framework.

1) COUNTER-BASED DDoS ATTACK DETECTION APPLICATION

Counter-based detection application depends on the counter values to measure the DDoS attack. This module is further divided into two sub-modules to monitor and analyze the DDoS attack in the SD-IoT network. C-DAD consists of different algorithms that are based on counter variables to help in detection. C-DAD is further elaborated in the counter-based attack detection module.

2) ATTACK MITIGATION MODULE/APPLICATION

The attack mitigation module gets the reported malicious flow from the counter-based detection module and performs the countermeasure actions.

B. CONTROL LAYER

In the SDN paradigm, control layer enables router functionalities at the central point, and it is dynamically programmable. This layer's main function is to control and manage the IoT network through dynamic, programmable, and customized fashion. This layer consists of two components, i.e., the SDNWISE controller and IoT controller.

1) SDNWISE CONTROLLER

SDN controller is the brain of SDN network. It defines policies and manages flow control of SDN based IoT network. In this research, we use the SDNWISE controller for the Internet of Things devices, which needs low power for their functionality. Moreover, the controller is the central point where the whole network is managed and controlled, such as network policy, flow table control. It exposes the API that enables the developer to write an application for the IoT network.

2) IoT CONTROLLER

IoT controller is the mediator between the SD-IoT network and the SDNWISE controller. It receives the SDNWISE controller's traffic and converts it into an appropriate format, which is understandable to the SD-IoT network. It also performs the same action vice-versa. In this research, we use only one type of network, but the IoT controller plays a significant role in extending work on the heterogeneous network. In a heterogeneous network, and IoT controller has added, which will operate according to that particular type of network. Multiple IoT networks can be connected with SDNWISE by just adding an IoT controller. This research focuses on one type of network just because we wanted to narrow down our problem with defined scope. In this article C-DAD only works on IEEE 802.11.4 protocol, but for future research direction, we will be working with multiple IoT networks in heterogeneous environments.

C. INFRASTRUCTURE LAYER

This layer is used only to forward the packets and is also called a data forwarding plane. According to SDN paradigms, this layer includes forwarding devices such as OpenFlow switches, which are not intelligent and hence forward the traffic according to the flow table entry. Sensor OpenFlow Switch (SOFS) acts as an OpenFlow switch to forward the traffic between IoT nodes in this research. There two types of devices are used in this layer, like SOFS and IoT nodes.

1) SENSOR OPENFLOW SWITCH

Sensor OpenFlow Switch is customized as an OpenFlow-based switch, which forwards the IoT traffic. Mostly SOFS acts as a forwarding device and less intelligent. In this research, we programmed it to report traffic load to the controller. It also triggered an alert message to the attack receiver component in the SDNWISE controller via IoT gateway, when traffic load reaches the threshold value. It also reset the threshold counter upon receiving the message from attack receiver components through the SDN controller, since the reported traffic load was not malicious.

2) IoT NODES

These are tiny Low powered-devices, which are based on the 6LowPAN protocol to communicate with each other. These devices spread in open space. Therefore, they can be easily compromised by an attacker. In this research, the attacker compromises IoT devices and generates the flooded traffic to perform the DDoS attack in the IoT network.

D. COUNTER BASED ATTACK DETECTION APPLICATION

The Counter based Attack Detection (C-DAD) uses the Counter values based on the IoT network's current status and its appropriate flows to detect malicious activity in the SD-IoT network. C-DAD detects the DDoS or any anomalies in the SD-IoT network, which depends on the counter algorithm with different parameters to measure IoT traffic statistics. Furthermore, it is divided into four sub-modules, which interact with others to perform the attack detection operation. The malicious flow will report to the mitigation module. The Counter-based Attack Detection Module is

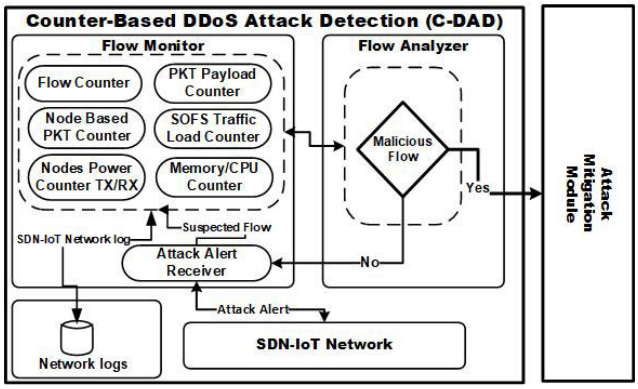


FIGURE 4. Counter-based detection module.

further divided into two sub-modules; Flow Monitor and Flow Analyzer, as shown in Figure. 4. The overall steps performed in C-DAD are shown in Algorithm 1.

1) FLOW MONITOR

The Flow Monitor consists of different counter-based algorithms; which are running to monitor the network status. These algorithms collect the information from the network log buffer for monitoring the network. It will monitor real-time data flows. Whenever any malicious activity occurs, it will collect the counter information of different algorithms and forward it to the flow analyzer component. It will detect malicious activity using the current status of network devices and their flows. Various algorithms are running in the module, getting different IoT network statistics like node traffic, SOFS Traffic, and IoT network bandwidth to monitor anomalies in the SD IoT network.

This sub-module based on a counter-based algorithm is used to monitor the IoT network status. If it finds any malicious behavior in the network, it runs the C-DAD algorithm to detect suspicious activity.

- **Flow Counter.** It counts the number of flow entries to monitor the Newflow attack, which occurs due to spoofed addresses of flows. This algorithm detects the Newflow attack, in which the attacker tries to spoof the source IP address of the IoT network and dynamically change that address so that it will cause the overflow in the flow table.
- **Packet Payload Counter.** It detects the attacks in which an attacker injects the malicious scripts in the packet's payload. These scripts perform the illegal activities at the receiving end. Also, this algorithm counts packets with the same size of payloads. It monitors counter value to reach the threshold. Whenever it's value gets the threshold value, it will generate an alert to the flow Analyzer. The flow Analyzer will further analyze the SD-IoT network with respect to other algorithms. If it found anomalies in the result, it will notify to Attack Mitigation module.
- **Node-Based Packet Counter.** In the IoT network, each node receives the packet in the normal fashion. It monitors the packet status of each node to detect any anomaly in the network. This algorithm counts the sending and receiving packets status of each node to determine any network threat. If found, it will report to Flow Analyzer. The first research experiment is based on the number of packets and used as counter values for the Packet-based threshold value, which is considered a threshold value and will be considered in Example 1's experiments.
- **Node Transmission/Receiving Power.** It monitors the power ratio of each node in the IoT network. Power is one of the main essential features of any IoT device because sometimes these devices are deployed in the war zone or earthquake or any other surveillance application. The attacker tries to attack these nodes to consume the powers and target bursty traffic to consume power. The algorithm also detects any anomaly in the power of nodes that is an alarming condition for the network so that it will inform to Flow Analyzer for further action.
- **SOFS Traffic Load Counter.** This algorithm determines the traffic load on the IoT network. It also monitors in/out traffic load on sensor OpenFlow switch per second, which is beneficial for the network's

Algorithm 1 C-DAD Counter-Based DDoS Attack Detection

Step 1: Test the Network status call AttackTest
Step 2: set packets =pkl=1000
set payload =pl=1000*Nodes
set SDFSl=3
set $FlowlFl = SDN_WISE_RLS_MAX$
set DDOS limit DDOSl=[fl and pl]
Step 3: Run Detection Module
Step 3.1: Compute pk=computepk() **GOTO Step 5**
Step 3.2: Compute p=computep() **GOTO Step 6**
Step 3.3: Compute flow=computeflow() **GOTO Step 7**
Step 3.4: Compute SDFS=computeSDFS() **GOTO Step 8**
Step 3.5: Compute DDOS=computeDDOS() **GOTO Step 9**
Step 4: Check The DDoS Attack
Step 4.1: If (pk>pkl) OR (p>pl) OR (SDFS > SDFSl) OR (F>Fl) OR (DDOS>DDOSl)
Step 4.2: Print Message Network is compromised
Step 4.3: Else-If set (attack==false)?true
Step 5: computepk():for p in DataPackets:
Packets[p.source]++
Packets[p.destination]++
return max(Packets)
Step 6: computep():for p in NetworkPackets:
$(p.type = SDN_WISE_DATA)counter++$
return counter
Step 7: computeflow():
Step 8: computeSDFS():
Step 9: computeDDOS():for s in NetworkPackets.title:
(s.contains('DDoS')) ddos++

real-time workload. It also provides traffic data that will help calculate the throughput, through which it becomes easy to determine the status of the IoT network. It will report to Flow Analyzer if it finds any anomaly in the SD-IoT network's traffic flows. This algorithm also calculates the volume of traffic transmit received in IoT Network, through which it is easy to detect DDoS attacks. The second and third examples are conducted with bandwidth parameters used as threshold values such as 0.5K bytes per second and 1K bytes per second.

2) FLOW ANALYZER

Flow Analyzer will run the algorithm on that flow to further analyze the data flow. Whenever it receives an alert from the flow monitoring module, it will gather the counter values from other algorithms to analyze their values and decides whether the network is under attack or not. After combining all results, it will decide to identify the malicious traffic. If malicious traffic is found, then it will notify to Mitigation module, which will perform appropriate action to countermeasure the attack. Whenever any alert regarding suspicious flow is received, it will run the C-DAD and get the result, which shows that the IoT network is running safely without any malicious activity. It will then inform back to IoT Network via Attack Alert Receiver to reset the Threshold Counter. Flow Analyzer runs the (C-DAD) Counter-based DDoS Attack Detection Module, which consists of different sub-modules in the algorithm as can be seen in Algorithm 1 that works together to detect the attack.

3) ATTACK ALERT RECEIVER

It receives the alert message regarding suspicious traffic from the SD-IoT network via the IoT controller to inform the Flow Monitor. Flow Analyzer also sends the message to it. If no attack is found in the reported flows, it notifies the Flow monitor and threshold counter of the reported algorithm will be reset.

4) NETWORK LOG

SD-IoT network counter information is stored in the network log buffer. Each node of the SD-IoT network and sensor OpenFlow switch forward the network counter information in the network stream to write network log buffer at SDNWISE controller via IoT controller. Flow Monitor's components are fetching the network counter information to monitor any malicious flow in the SD-IoT network. The flow monitor receives an attack alert from sensor OpenFlow switch via Attack Alert Receiver. The Flow Analyzer also gets counter values of algorithms from the network log buffer via Flow Monitor to detect the malicious traffic in the SD-IoT Network.

E. SENSOR OPENFLOW SWITCH AND IoT NODE

IoT nodes are scattered in clusters, and each cluster has a head node with sufficient resources to communicate with IoT controller. IoT node, with sufficient resources, will implement SDN techniques and play the role of OpenFlow switch.

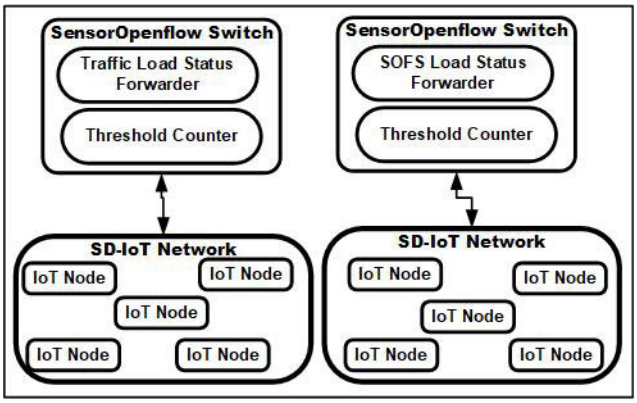


FIGURE 5. SD-IoT-based nodes and sensor Openflow switch.

1) SENSOR OPENFLOW SWITCH (SOFS)

SD-IoT network support customized sensor OpenFlow switch and forwards the IoT traffic based on rule and policy-related towards appropriate flow, which is received from the SDNWISE controller.

SOFS collects the statistics about IoT network traffic and forwards it to the SDNWISE controller, who writes the counter information regarding the SD-IoT network in the network log buffer. Two algorithms are running on the SOFS to help regarding malicious flow in the SD-IoT network, as depicted in Figure. 5.

- **Traffic Load Forwarder**. The volume of traffic load on the sensor OpenFlow switch describes the traffic In and Out to SOFS, and it displays the average traffic throughput. These values are forwarded to the C-DAD algorithm for further traffic analysis.
- **Threshold Counter**. This algorithm monitors the IoT network traffic with respect to threshold values. If the threshold counter reaches the threshold value, it generates an alert to the SDNWISE controller via the IoT controller for further action.

2) IoT NODES

IoT nodes will generate a different type of traffic in the SD-IoT network. In this simulation, we generate two types of traffic with the appropriate type of algorithm, such as normal traffic, generated by IoT nodes relevant to the particular application deployed. A different kind of traffic is generated through malicious IoT nodes to simulate the DDoS attack detection module with the help of SDN.

F. ATTACK MITIGATION MODULE

C-DAD analyzes the IoT traffic and detects DDoS attacks, anomalies, and threats. This module performs the countermeasure action on malicious flow, which is reported by C-DAD. This module has sub-modules, which will coordinate with each other to mitigate attacks. IoT Attacks Mitigation

Module will exploit SDN security features to mitigate DDOS and intrusion in IoT. The below-given sub-modules are used during the attack mitigation process.

1) MALICIOUS FLOW ENTRY

In this sub-module, the malicious flow will be added to the SD-IoT network.

2) MALICIOUS NODE REMOVE

SDNWISE controller will remove the malicious node from the SD-IoT network, which generates malicious traffic using network graphic API.

V. EXPERIMENT EVALUATION AND RESULTS

This section will briefly describe the experiments performed to detect DDoS attacks in the SD-IoT network and analyze the different parameters' results. This research analyzes the DDoS detection method based on Counter-based attack detection application by using the SDN paradigm. The simulation-based experiments are conducted on SDNWISE controller [21], [45] and Cooja simulator [46], [47], with the help of the Contiki operating system. The hardware configurations of the experimental environment for evaluating the C-DAD performance are as follows: Ubuntu 16.04 LTS Core i7 6850K @ 3.60 Ghz processor, 16 GB RAM with GPU supported machine. The experiments are simulated by using a Cooja simulator with the help of the Contiki OS. Cooja simulator helps to create a realistic virtual SD-IoT network, which consists SDNWISE controller, SOFS and IoT nodes.

A. SDNWISE-BASED IoT NETWORK SETUP

The experiments are conducted with different attacking nodes concerning parameters such as burst, packet payload, and others to launch the DDoS attack in the SD-IoT network. Firstly, we generate malicious traffic with different frequencies and with varying sizes of the packet's payload, along with the normal traffic. A basic SD-IoT network topology is shown in Figure. 6. The traffic status is being monitored; whenever it exceeds the threshold value, the attack alert triggers the alert to the controller. The C-DAD will run the Flow Monitor and Flow Analyzer sub-modules based on the counter-based detection algorithm to detect a DDoS attack.

We conducted three experiments with varying parameter such as IoT nodes, attack nodes, packet payload, packet burst, and others as shown in Table 2. In each experiment, these same parameters are used with different topology.

B. EXPERIMENTAL SETUP

Each experiment has different parameters, so we use one parameter as a variable and others as constant to find out the other parameters. In this case, we experimented with two different case studies. Each case study has evaluated three parameters, one by one, with various experiments to find other attributes. Our experiments are based on variations of the following attributes.

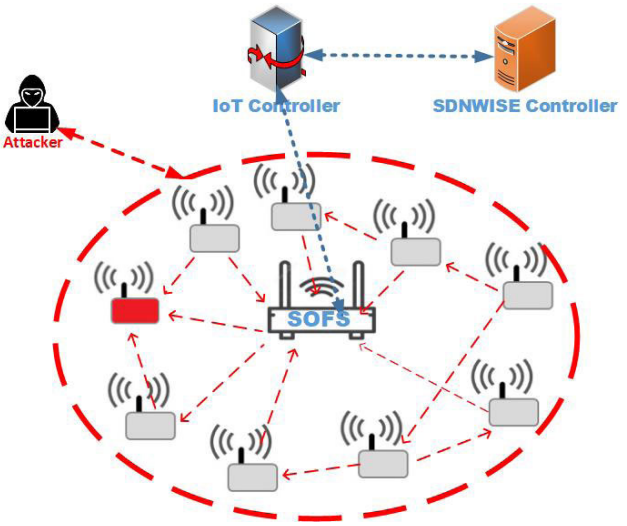


FIGURE 6. SD-IoT network affected with DDoS.

- **Number of IoT Nodes or Normal Nodes**.
- **Number of Attack Nodes**
- **Packet Payload**.
- **Packet Frequency (Normal and Burst Mode)**.

All the SD-IoT network parameters will vary for a particular network environment, just like a smart home network has different values than a smart city network. It also depends on the network size, traffic type, and nature of the system's applications.

- **Total Data Packet**. *Total number of data packets captured in IoT network.*
- **Malicious Packet**. *Malicious packet detected by controller.*
- **Normal Packet**. *Normal IoT network traffic.*
- **SDNWISE Controller Workload (KB/Sec)**. *Number of traffic processed by controller during attack detection.*
- **SD-IoT Network Throughput (%)**. *The amount of normal traffic processed by SOFS represents the usage of network traffic of an IoT network in percentage. Whenever an IoT network is under attack and without attack, throughput measured is a hundred percent.*
- **CPU Utilization (%)**. *Attack detection application uses CPU during attack detection.*
- **Allocated Memory(MBytes)**. *Total Memory allocated to attack detection application.*
- **Memory Utilization (MB)**.*The memory used by application.*
- **Attack Detection Time (Seconds)**. *Time to detect the DDoS attack in IoT network with the help of counter-based detection module which is running at the top of SDNWISE controller.*

1) EXPERIMENTS THRESHOLD VALUE

In this research, C-DAD algorithms decides to run the attack detection algorithm in a specific condition. These conditions are based on the number of suspected packet received and the workload of the SD-IoT network

TABLE 2. Experiment No.1.1 Variation in the number of IoT nodes.

EXP NO	IoT Nodes	Packet Payload (bytes)	Burst Mode Frequency (Packet/Sec)	Attack Nodes	Payload Counter	Flow Table Overflow	SDNWISE Controller Workload KB/Sec	CPU Utilization (%)	Allocated Memory	Memory Utilization (MB)	Attack Detection Time (Seconds)	Total Time (Seconds)
1	5	30	0	0	0	0	1.4609	3	139	29	0	2
2	10	30	0	0	0	0	2.0224	4	170	38	0	2
3	15	30	0	0	0	0	2.373	5	120	70	0	2
4	20	30	0	0	0	0	2.4218	7	218	162	0	2
5	30	30	0	0	0	0	2.623	9	304	218	0	2

exceeds normal behavior. After the SD-IoT network reached a selected particular threshold value, the SOFS generates the alert and runs the C-DAD algorithm to detect DDoS attacks. For this study, there are three experiments conducted in this regard. Experiment 1 is based on the packet-based threshold value, and Experiment 2 and 3 are based on bandwidth values.

- **Packet-based**. This threshold value is calculated with different experiments to find an appropriate threshold value. This threshold value is used in a bursty SD-IoT network. The network received one thousand suspected packets that might have a similar payload size. One node received many packets; whenever these conditions meet in the SD-IoT network, the SDNWISE controller runs the C-DAD algorithm.
- **Bandwidth-based**. The experiments are conducted with a bandwidth-based threshold, in which the SDNWISE controller runs the C-DAD algorithm when the SD-IoT network bandwidth reached at 0.5 KBytes (Experiment 2) or 1 Kbytes/sec (Experiment 3).

These threshold values are decided very carefully. We run the SD-IoT network with maximum nodes and investigates the minimum and maximum bounds of both packet-based and bandwidth-based threshold values. The minimum value threshold might overhead on the controller every time we run the algorithm for legitimate traffic. If we set the threshold with maximum value, we may run the C-DAD algorithm lately, and the attack might be earlier before the counter reaches the threshold value.

C. EXPERIMENT NO. 1

In this scenario, the experiment is conducted on the SD-IoT network infrastructure with bursty traffic behavior. In this network, the IoT node transmits one message per second. The worst-case message received at a single node in the network with thirty nodes is 29 messages. This experiment is conducted based on a smart system-based scenario in which network traffic is generated in bursty mode from the smart objects. Meanwhile, the SD-IoT network mostly has tiny devices with very low traffic, but these devices are very large in numbers, so attackers can easily target such networks. In this bursty behavior-based network, we assume the threshold value of SD-Network. If it receives one thousand suspected packets, it generates an alert to check network status and run the C-DAD algorithm. In this scenario, five different experiments are conducted for each parameter like attack nodes, burst mode, packet payload size, normal network traffic, or without attack. Each parameter is separately discussed in the following subsections.

1) NORMAL NODES

These experiments are conducted with different IoT nodes in numbers without malicious traffic; so, IoT network is having full respective throughput with IoT nodes. CPU Utilization increases by 1% with an increment of IoT node parameters and increases memory utilization with respect to IoT nodes, as shown in Table 2.

2) ATTACK NODES

In this scenario, the experiment is conducted with the different variable number of attack nodes with other fixed parameters such as burst frequency, packet payload, and IoT nodes. This experiment has used the number of packets as a threshold value instead of bandwidth or SDNWISE controller workload. The packet-based threshold experiment with a minimum burst of 510 packets/sec results in a high amount of traffic in the SD-IoT network compared to the other two experiments based on the bandwidth-based threshold. The attack nodes parameter plays a significant impact on IoT Network throughput compared to the other two experiments, i.e., 2 and 3. The experiment is conducted with the four attack nodes. The SD-IoT network throughput decreases approximately to 75%, and a 92% decrease is observed with eight attack nodes, as shown in Table 3 and Figure. 7(c). We know that this experiment is conducted with a four-time large (510 packet per second) value of Burst Frequency towards the other two experiments, 120 Packet per second. That's why CPU utilization has been increased than Experiment 2 and 3. But, in the throughput-based threshold value, CPU utilization decreased from 10 to 8, and in the Packet-based threshold value, the CPU Utilization has increased on a few points. It has been decreased later, but it is still larger than the highest value of CPU Utilization in experiments 2 and 3. The attack detection time decreased from the high value to low with respective attack nodes. These values increase slightly because the burst frequency values are greater than with one shown in experiment 3. But there is no significant impact on packet threshold value using attack nodes.

3) BURST MODE (PACKET/SEC)

The experiments are conducted with the burst frequency parameter as a variable and other constant parameters such

TABLE 3. Experiment No.1.2 Variation in the number of the attack nodes.

EXP NO	IoT Nodes	Packet Payload (bytes)	Brust Packet/Sec)	Attack Nodes	Total Data Packet	Malicious Traffic	Normal Traffic	SD-IoT Network Throughput	SDNWISE Controller Workload KB/Sec	CPU Utilization (%)	Allocated Memory	Memory Utilization (MB)	Attack Detection Time (Second)
1	30	30	510	4	454	334	120	26.43	10.172852	15	379	230	140.826
2	30	30	510	8	364	334	30	8.24	11.103516	18	419	332	51.969
3	30	30	510	12	349	334	15	4.30	6.4746094	21	461	347	36.662
4	30	30	510	15	343	334	9	2.62	5.7421875	14	505	468	30.648
5	30	30	510	20	339	334	5	1.47	8.62793	16	501	330	26.441

TABLE 4. Experiment No.1.3 Variation in the burst frequency of the attack nodes.

EXP NO	IoT Nodes	Packet Payload (bytes)	Burst Frequency (Packet/Sec)	Attack Nodes	Total Packet	Malicious Packet	Normal PKT	SD-IoT Network Through-put(%)	SDNWISE Controller Workload KB/Sec	CPU Utilization (%)	Allocated Memory	Memory Utilization (MB)	Attack Detection Time (Seconds)
1	30	30	510	10	352	334	18	5.11	6.430664	14	377	237	41.723
2	30	30	1050	10	344	334	10	2.91	7.3095703	15	305	229	23.118
3	30	30	1500	10	340	334	6	1.76	5.2001953	18	504	289	14.133
4	30	30	2100	10	340	334	6	1.76	5.5078125	14	378	217	12.466
5	30	30	3000	10	338	334	4	1.18	7.7490234	16	318	260	6.641
6	30	30	4050	10	337	334	3	0.89	9.008789	14	378	139	6.739

TABLE 5. Experiment No.1.4 Variation in the packet payload.

EXP NO	IoT Nodes	Packet Payload (bytes)	Burst Frequency (Packet/Sec)	Attack Nodes	Total Packet	Malicious Packet	Normal PKT	SD-IoT Network Through-put(%)	SDNWISE Controller Workload KB/Sec	CPU Utilization (%)	Allocated Memory	Memory Utilization (MB)	Attack Detection Time (Seconds)
1	30	30	510	15	343	334	9	2.62	7.5214844	20	380	253	30.424
2	30	37	510	15	343	333	10	2.92	9.789551	22	502	352	30.27
3	30	47	510	15	343	333	10	2.92	10.997559	21	507	457	30.363
4	30	67	510	15	343	333	10	2.92	19.85254	38	507	481	28.94
5	30	87	510	15	343	333	10	2.92	31.21289	40	502	492	28.659

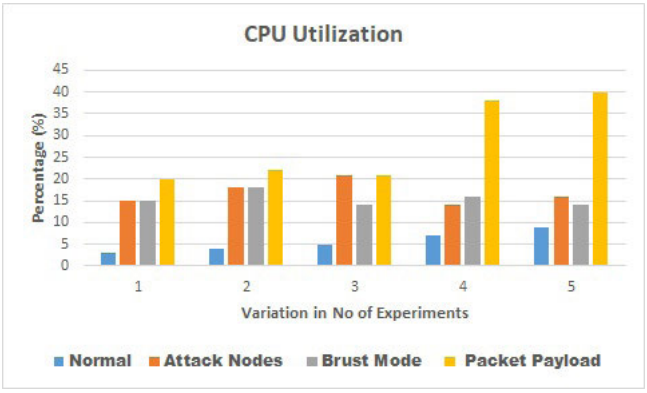
as; attack nodes, packet payload, and IoT nodes. There is a very high impact of burst Frequency parameter on SD-IoT network throughput, which is considered higher than the attack node parameter. The experiment with minimum burst size decreases 95% of SD-IoT network throughput, and when burst frequency is doubled than the previous value of the SD-IoT network, the throughput decreases approximately to 98%. Additional results are shown in Table 4 and the SD-IoT network throughput Figure. 7(c). Moreover, the CPU utilization has been increased mostly with the same ratio in previous parameters' attack nodes, a 1% increment each time by changing burst size. But with the largest burst size, the CPU utilization fluctuates high and low depending on the flooding burst of packets received, so, it becomes high otherwise low as shown in Table 4 as well as in the CPU Utilization Figure. 7(a). Attack detection time increases compared to the previous parameter attack node from 140 seconds, with four attack nodes to 42 seconds with ten attack nodes. But when we compare the same burst configuration with the same attack node, no big difference is found. Meanwhile, the attack detection time decreases as Burst Frequency increases compared to the increased ratio of attack nodes; Burst Frequency has a high impact on attack detection time. Memory utilization fluctuates high and low with respect to Burst Frequency. If the flooding ratio of malicious traffic is high with normal traffic, the algorithm might take high memory and on the other hand, take low if the flooding ratio of malicious traffic is low.

4) PACKET PAYLOAD

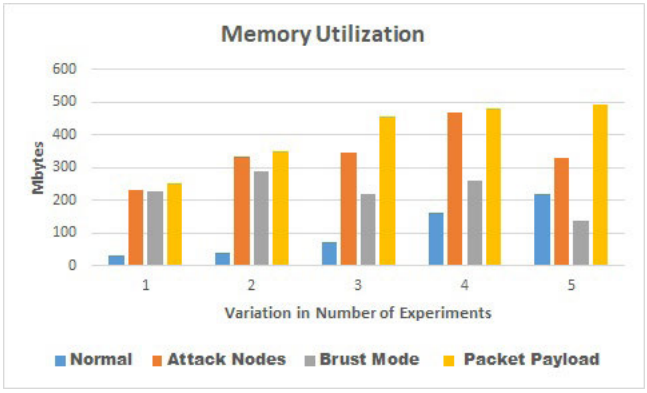
These experiments were conducted with the packet payload variable as a parameter with other fixed parameters as defined above. We conducted experiments with five different packet payloads, but there is no significant impact on the SD-IoT network throughput and attack detection time. However, the Packet Payload parameter has a significant impact on CPU utilization and also on memory utilization. The CPU utilization increased from 20 to 40 with different packet payloads, which are very high compared to all experiments conducted with a packet-based threshold and the two other experiments based on traffic-based threshold values such as throughput 0.5 and 1 KBytes per second. Another significant impact of packet payload on the SDNWISE controller workload is that it increases the value as compared to previous experiments with the respective example. Therefore, three parameters were affected with Packet Payload, such as CPU, Memory, and SDNWISE controller workload, as shown in respective Figure. 7(a), (b), and (d) of each parameter along with the results given in Table 5.

D. EXPERIMENT NO. 2

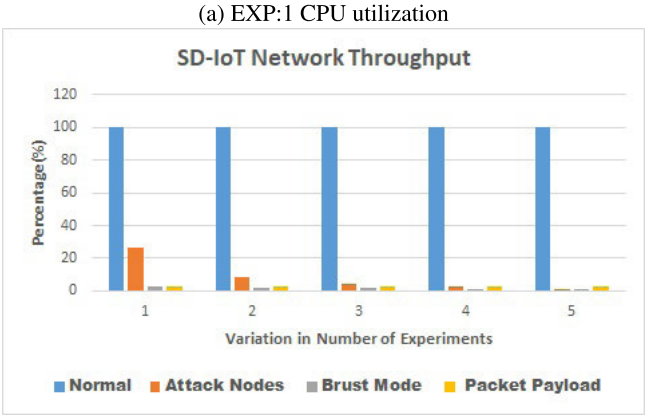
In this scenario, we conducted experiments that are based on the smart home system. This experiment's network traffic model assumes that each node transmits one message per second; in worse-case, each node receives (n-1) messages. In this experiment, C-DAD behavior is based on bandwidth to detect the DDoS attack; the threshold value is calculated according to smart home network traffic with 30 nodes at the maximum threshold value. The SOPFS generates the SDNWISE controller's alert if SD-IoT network traffic bandwidth is reached at 0.5 KBytes/second. The controller runs the C-DAD algorithms to check the network status and generates a message about any anomalies in the SD-IoT network. This section



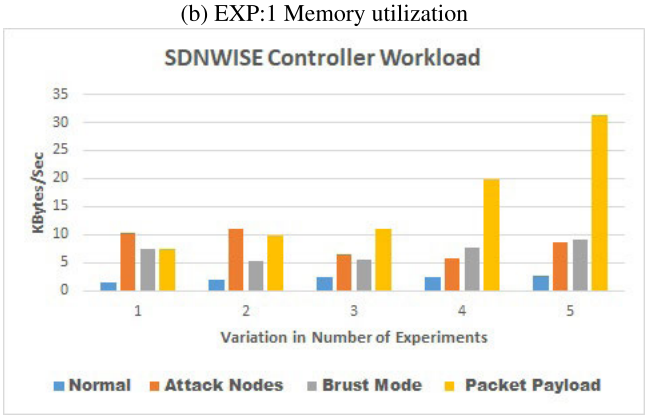
(a) EXP:1 CPU utilization



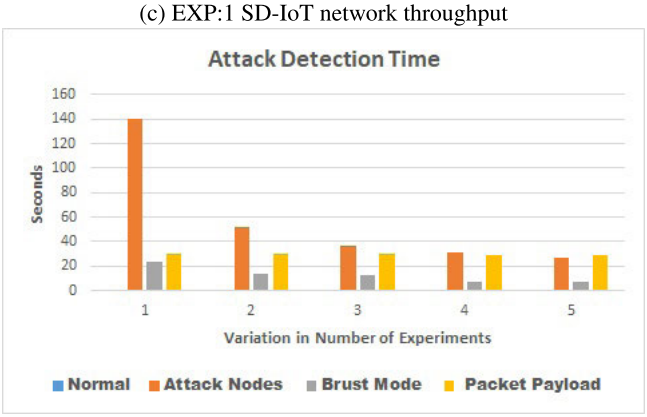
(b) EXP:1 Memory utilization



(c) EXP:1 SD-IoT network throughput



(d) EXP:1 SDNWISE controller workload



(e) EXP:1 Attack detection time

EXPERIMENT	Normal (Nodes)	Attack Nodes	Brust Mode (Packet/Second)	Packet Payload(Bytes)
1	5	4	1050	30
2	10	8	1500	37
3	15	12	2100	47
4	20	15	3000	67
5	30	20	4050	87

(f) EXP:1 Experiment's parameters values

FIGURE 7. Experiment 1 with packet-based threshold.

evaluates the results based on the experiment conducted with 0.5 Kbytes/sec threshold value and other parameters, as mentioned in the above experiment.

1) NORMAL NODES

The experiments were conducted in an SD-IoT network with different numbers of IoT nodes having distinct topology. In this scenario, only one IoT node or IoT node parameter has been changed each time without attack traffic. CPU and memory utilization has been increased with respect to IoT nodes, and SDNWISE controller workload increased, as shown in Table 6. There was no attack traffic; therefore, the attack detection time was zero, and the network was with 100 percent throughput of IoT network traffic.

2) ATTACK NODES

The first experiment was performed with four attack nodes and other constant parameters such as burst mode/frequency, packet payload, and IoT nodes. We observed around 60% SD-IoT network throughput, and out of that, 40 % throughput was consumed by malicious traffic, whereas, the attack was detected in 45 seconds, as shown in Figure. 8(c) and (e). The other experiments are conducted by changing the attack nodes, and in that scenario, we noticed that the SD-IoT

TABLE 6. Experiment No.2.1 Variation in the number of IoT nodes.

EXP NO	IoT Nodes	Packet Payload (bytes)	Burst Mode Frequency (Packet/Sec)	Attack Nodes	Total Packet	Malicious Packet	Normal PKT	IoT Network Through-put(%)	SDNWISE controller Workload KB/Sec	CPU Utilization (%)	Allocated Memory	Memory Utilization (MB)	Attack Detection Time (Seconds)
1	5	30	0	0	140	0	140	100.00	0.068359375	3	139	29	0
2	10	30	0	0	289	0	289	100.00	0.07128906	4	170	38	0
3	15	30	0	0	433	0	433	100.00	0.059570312	5	120	70	0
4	20	30	0	0	510	0	510	100.00	0.11230469	7	218	162	0
5	30	30	0	0	833	0	833	100.00	0.22753906	9	304	218	0

TABLE 7. Experiment No.2.2 Variation in the number of the attack nodes.

EXP NO	IoT Nodes	Packet Payload (bytes)	Burst Mode Frequency (Packet/Sec)	Attack Nodes	Total Packet	Malicious Packet	Normal PKT	IoT Network Through-put(%)	SDNWISE controller Workload KB/Sec	CPU Utilization (%)	Allocated Memory	Memory Utilization (MB)	Attack Detection Time (Seconds)
1	30	30	120	4	70	28	42	60	0.5546875	10	207	65	45
2	30	30	120	8	20	16	4	20	0.5126953	11	208	95	5
3	30	30	120	12	23	20	3	13.04347826	0.52734375	11	148	118	5
4	30	30	120	15	36	32	4	11.11111111	0.57128906	11	209	145	9
5	30	30	120	20	30	28	2	6.666666667	0.5566406	11	206	133	8

TABLE 8. Experiment No.2.3 Burst frequency of the attack nodes.

EXP NO	IoT Nodes	Packet Payload (bytes)	Burst Mode Frequency (Packet/Sec)	Attack Nodes	Total Packet	Malicious Packet	Normal PKT	SD-IoT Network Through-put(%)	SDNWISE controller Workload KB/Sec	CPU Utilization (%)	Allocated Memory	Memory Utilization (MB)	Attack Detection Time (Seconds)
1	30	30	120	10	89	60	29	32.58	0.5839844	10	204	59	40.925
2	30	30	240	10	11	8	3	27.27	0.5126953	12	148	75	0.98
3	30	30	480	10	19	16	3	15.79	0.6591797	13	149	84	0.89
4	30	30	960	10	35	32	3	8.57	1.2890625	14	206	79	0.85
5	30	30	1920	10	67	64	3	4.48	2.5488281	17	214	119	0.416
6	30	30	3840	10	131	128	3	2.29	5.1123047	18	205	108	0.089

network throughput decreased from 60% to 20% and similarly attack detection time from 45 seconds to 5 seconds. After eight attack nodes, we have further increased the attack node parameter; there is no such high impact on CPU utilization and attack detection time, but the memory utilization has been increased, and SD-IoT network throughput decreases with respect to increase of the attack nodes as shown in Table 7 and respective Figure. 8(a), (b), (c), and (e).

3) BURST MODE (PACKET/SEC)

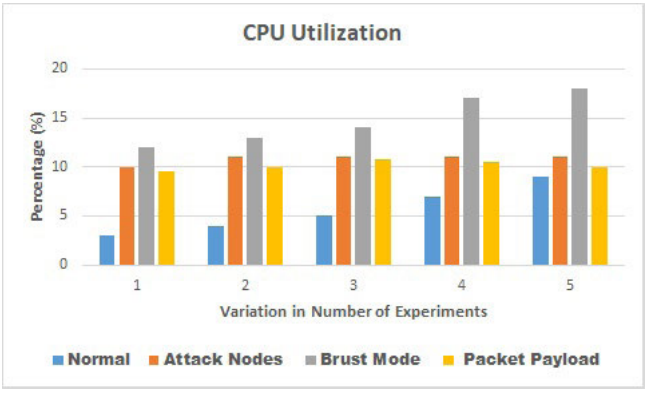
The burst frequency parameter has a high impact on the attack detection time as well as network throughput. At an initial experiment with burst frequency 120, which is the minimum level set to other experiments, such as attack nodes, the attack detection time is almost the same as the initial experiment of attack node with 4. But the SD-IoT network throughput decreased to 32% with a fixed value of other parameters, such as packet payload and attack nodes. However, when we double the burst frequency of the previous experiment, the attack detection time shoots down from 40 seconds to approximately 1 second, as shown in Figure. 8(e). The CPU Utilization increased 1% with an increase in the Burst frequency, and memory utilization increases with the same ratio as we increase the burst frequency. The SD-IoT network throughput decreased approximately half of its previous from the first experiment in which it was 32 percent. Another important aspect of this parameter as compared to other parameters is that the SDNWISE controller workload increased twice from its previous experiments because these values increased rapidly from its 0.5 Kbytes/sec value of the threshold value, as shown in Table 8 and Figure. 8(d). It means after 0.5 Kbytes/sec threshold, the attack detection algorithms run and detect the attack. During that period, the SDNWISE controller workload has increased very fast as compared to other parameters. However, initially, the SDNWISE controller workload increased at very low rate from the threshold values.

4) PACKET PAYLOAD

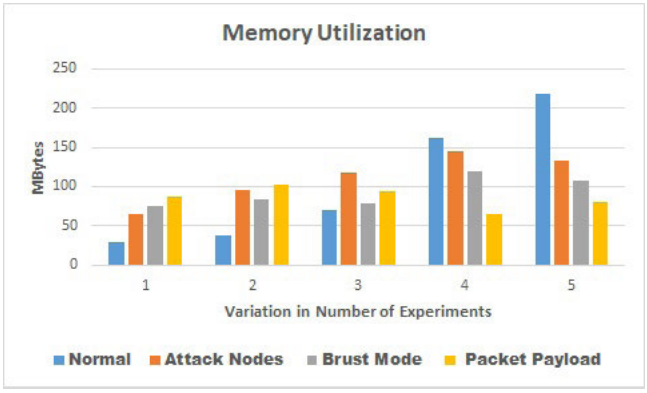
These experiments were conducted with different packet payloads and other fixed parameters such as burst frequency, attack nodes, and the number of IoT nodes. In this scenario, the results of experiments affect two different points. First, if the packet payload is small between 20 bytes to 40 bytes, the SD-IoT network throughput is approximately 30% to 35%, and memory utilization stands at 90 to 100 Mbytes. Secondly, if the packet payload is from 50 bytes to 90 bytes, the SD-IoT network throughput mostly becomes constant at 57%, and the attack detection time is approximately 1 second. There is no such effect on CPU utilization, which was only 9% to 10% with all experiments deploying different packet payloads, as shown in Table 9 and Figure. 8(a), (b), (c),and (e). These are two different types of results due to a threshold value, which is taken as a parameter that is based on SD-IoT network traffic bandwidth.

E. EXPERIMENT NO. 3

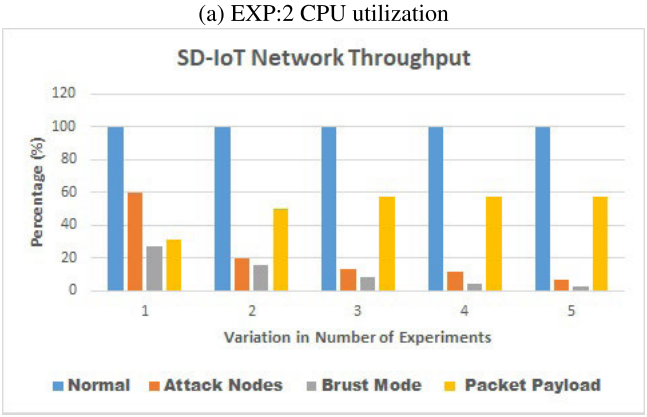
In this scenario, we increase the threshold value to double from 0.5 KBytes/sec to 1 KBytes/sec, and all other parameters are the same which were used in experiment 2. The SOFS triggers the alert to run the Attack Flow Analyzer sub-module and determines whether the SD-IoT network is compromised



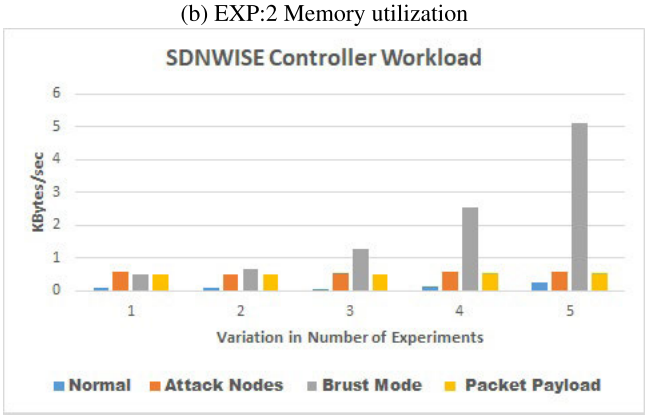
(a) EXP:2 CPU utilization



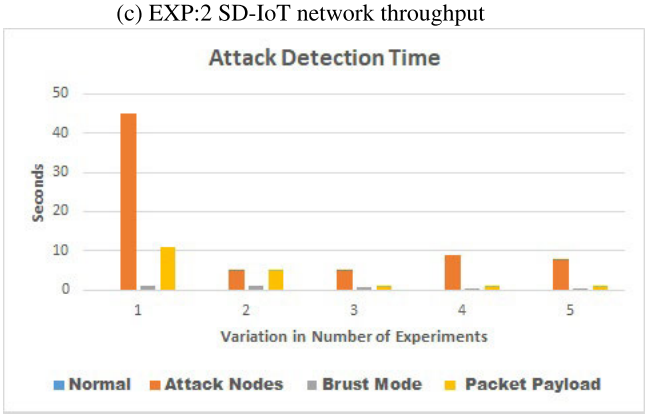
(b) EXP:2 Memory utilization



(c) EXP:2 SD-IoT network throughput



(d) EXP:2 SDNWISE controller workload



(e) EXP:2 Attack detection time

EXPERIMENT	Normal (Nodes)	Attack Nodes	Brust Mode (Packet/Second)	Packet Payload(Bytes)
1	5	4	240	30
2	10	8	480	40
3	15	12	960	50
4	20	15	1920	70
5	30	20	3840	90

(f) EXP:2 Experiment's parameters values

FIGURE 8. Experiment 2 with bandwidth-based threshold (0.5KBytes/Sec).

TABLE 9. Experiment No.2.4 Variation in the packet payload.

EXP NO	IoT Nodes	Packet Payload (bytes)	Burst Frequency (Packet/Sec)	Attack Nodes	Total Packet	Malicious Packet	Normal PKT	SD-IoT Network Through-put(%)	SDNWISE controller Workload KB/Sec	CPU Utilization (%)	Allocated Memory	Memory Utilization (MB)	Attack Detection Time (Seconds)
1	30	20	120	10	59	39	20	33.90	0.5361328	9	243	90	26
2	30	30	120	10	29	20	9	31.03	0.5126953	9.5	210	87	11
3	30	40	120	10	14	7	7	50.00	0.5078125	10	152	103	5
4	30	50	120	10	7	3	4	57.14	0.5078125	10.75	123	94	1
5	30	70	120	10	7	3	4	57.14	0.52246094	10.5	203	65	0.99
6	30	90	120	10	7	3	4	57.14	0.5419922	10	143	81	0.996

with a DDoS attack or not. The controller will achieve this by running the C-DAD algorithm. In this experiment, we also considered the same parameter, i.e., attack nodes, burst mode (packet frequency), and packet payload. All of these different experiments were conducted to get the different outcomes used to analyze the results. The SD-IoT network behavior is also the same for normal mode or network without attack, so we never include the same results and parameter values

TABLE 10. Experiment No.3.1 Variation in the number of the attack nodes.

EXP NO	IoT Nodes	Packet Payload (bytes)	Burst Mode Frequency (Packet/Sec)	Attack Nodes	Total Packet	Malicious Packet	Normal PKT	SD-IoT Network Through-put(%)	SDNWISE Controller Workload KB/Sec	CPU Utilization (%)	Allocated Memory	Memory Utilization (MB)	Attack Detection Time (Seconds)
1	30	30	120	4	179	72	107	59.77653631	1.109375	10	197	138	122.007
2	30	30	120	8	152	112	40	26.31578947	1.0185547	10	213	102	65
3	30	30	120	12	95	80	15	15.78947368	1.0126953	9	295	70	32.223
4	30	30	120	15	56	48	8	14.28571429	1.0605469	9	213	67	17
5	30	30	120	20	50	48	2	4	1.3691406	8	295	102	12.994

TABLE 11. Experiment No.3.2 Variation in the burst frequency of the attack nodes.

EXP NO	IoT Nodes	Packet Payload (bytes)	Burst Mode Frequency (Packet/Sec)	Attack Nodes	Total Packet	Malicious Packet	Normal PKT	SD-IoT Network Through-put(%)	SDNWISE Controller Workload KB/Sec	CPU Utilization (%)	Allocated Memory	Memory Utilization (MB)	Attack Detection Time (Seconds)
1	30	30	120	10	187	124	63	33.69	1.0664062	9	215	136	91.011
2	30	30	240	10	22	16	6	27.27	1.0546875	11	166	65	5
3	30	30	480	10	19	16	3	15.79	1.4355469	15	168	59	0.99
4	30	30	960	10	35	32	3	8.57	3.9697266	16	102	95	0.98
5	30	30	1920	10	67	64	3	4.48	4.92187	17	278	59	0.995
6	30	30	3840	10	131	128	3	2.29	35.390625	18	207	73	0.97

with the new table; refer to the previous Table 6 for the variations in IoT nodes or normal nodes.

1) ATTACK NODES

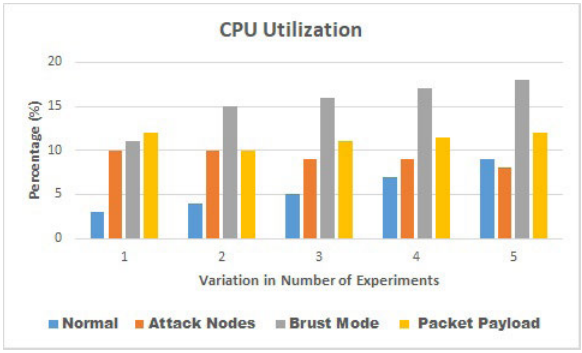
In this scenario, all experiments were conducted with different attack nodes, and other parameters are fixed, such as IoT nodes, burst sequence, and packet payload with their respective values as shown in the Table 10. In the experiment with the attack node parameter, the attack detection time is increased. It is decreased from high value towards low compared to the previous example with 0.5 Kbytes/sec threshold values, where it has been decreased from 45 to 5, but there is no such major difference in SD-IoT network throughput. The CPU utilization is decreased by 1% in each experiment by increasing the attack nodes. The memory utilization is initially high with the minimum attack nodes, and it has been decreased as attack nodes increased, however, in last with maximum attack nodes, it also increases. There is no effect on SDNWISE controller workload, but the SD-IoT network throughput decrease as we increase the attack nodes; the results are shown in Figure. 9(a),(b),(c),and (d).

2) BURST MODE (PACKET/SEC)

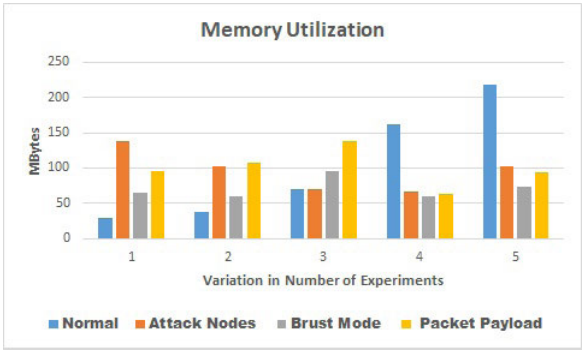
In the burst frequency parameter, the CPU utilization increased from 9% to 18%, by changing the burst value from 120 to 3840. Still, its behavior is opposite to the attack node parameter, where it has been decreased from 10% to 8%. Another difference is memory utilization, which has been increased initially. However, if burst frequency is doubled from an initial value, then memory utilization has been decreased, as shown in Table 11 and Figure. 9(a) and (b). But it sometimes fluctuates low to high due to burst frequency and the threshold value. If the condition where SDNWISE controller workload value is near to threshold value and at the same time, attack nodes generate the burst, it reaches quickly to threshold values, and the attack will be detected early. Therefore, it takes low memory if the SDNWISE controller workload is away from the threshold value, and there is no attack burst at that time. It takes high memory even with high burst value, because the threshold values are double than 1 Kbytes/sec compared to the previous experiment 2 in which threshold values were 0.5 bytes/sec. The attack detection time is high in the first experiment where burst frequency is 120, and in the second experiment, it becomes 5 seconds, with doubled burst frequency from the initial experiment. After the experiment was conducted with a burst frequency higher than 240, its behavior in attack detection time is the same as experiment 2, i.e., approximately 1 second.

3) PACKET PAYLOAD

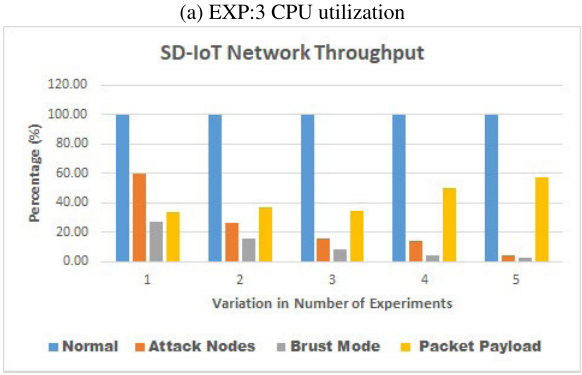
These experiments are conducted with the packet payload parameter as a variable with fixed parameters, such as attack node, burst frequency, and IoT node. With all these parameters, the SD-IoT network throughput is decreased from 100% to 34%, but after that, packet payload increased to 30, 40, and 50 bytes, and the SD-IoT network throughput increased a little bit with no big difference. In the last two experiments with packet payload 70 and 90 bytes, the SD-IoT network throughput increased from 34% to 50% and 57%, which shows that whenever the packet payload is 70 and above, the SD-IoT network throughput increases as compared to the previous experiments. The memory utilization has been increased from the initial one, but at some points where packet payload becomes 70 bytes, it decreased to 63 Mbytes and then increased to 94 Mbytes, as shown in Table 12 and Figure. 9(b), (c), and (e). The SDNWISE controller workload did not shoot up as it did with burst frequency from initial threshold values, after which alert of attack was generated. The attack detection time was initially high due to experiments conducted with a high threshold value compared to previous experiment 2. After that, as we increased packet payload by 10 bytes, the attack detection time decreased to half of its last values. It becomes 1 second with a packet payload size of 90 bytes, equal to the attack detection time



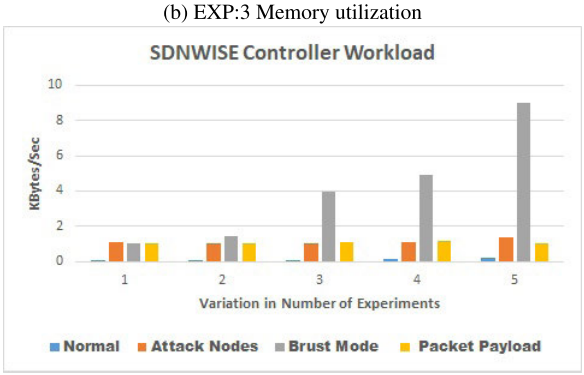
(a) EXP:3 CPU utilization



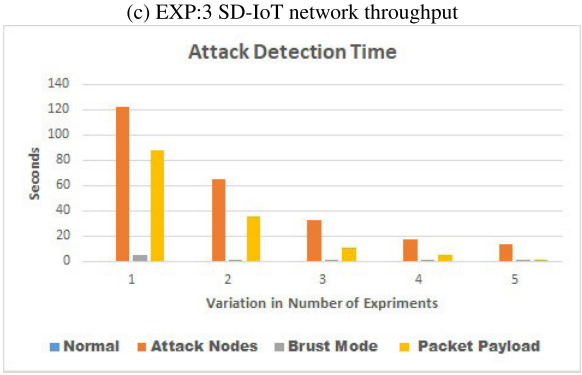
(b) EXP:3 Memory utilization



(c) EXP:3 SD-IoT network throughput



(d) EXP:3 SDNWISE controller workload



(e) EXP:3 Attack detection time

EXPERIMENT	Normal (Nodes)	Attack Nodes	Brust Mode (Packet/Second)	Packet Payload(Bytes)
1	5	4	240	30
2	10	8	480	40
3	15	12	960	50
4	20	15	1920	70
5	30	20	3840	90

(f) EXP:3 Experiment's parameters values

FIGURE 9. Experiment 3 with bandwidth-based threshold (1KBytes/Sec).

TABLE 12. Experiment No.3.3 Variation in the packet payload.

EXP NO	IoT Nodes	Packet Payload (bytes)	Burst Mode Frequency (Packet/Sec)	Attack Nodes	Total Packet	Malicious Packet	Normal PKT	SD-IoT Network Through-put(%)	SDNWISE Controller Workload KB/Sec	CPU Utilization (%)	Allocated Memory	Memory Utilization (MB)	Attack Detection Time (Seconds)
1	30	20	120	10	307	203	104	33.88	1.0410156	8	294	68	151.011
2	30	30	120	10	181	120	61	33.70	1.0087891	12	295	95	88
3	30	40	120	10	74	47	27	36.49	1.0537109	10	214	107	35
4	30	50	120	10	29	19	10	34.48	1.09375	11	213	138	10.997
5	30	70	120	10	14	7	7	50.00	1.171875	11.5	207	63	5
6	30	90	120	10	7	3	4	57.14	1.0253906	12	159	94	1

of the previous parameter for burst frequency's experiment with 240 burst frequency.

F. DISCUSSION

Each experiment was conducted to analyze SD-IoT network parameters and get a result to check the application's performance. The experiment consists of network parameters such as attack node, packet payload size, burst frequency, and the outcome parameters such as CPU and memory utilization, SD-IoT network throughput, SDNWISE controller workload, and attack detection time. In this subsection, we will discuss all the experiments conducted during this research to analyze the impact of each parameter with variable and constant values.

In the experiments concerning attack node, SD-IoT network throughput initially decreases 75% with four and 92%

with eight attack nodes, to the condition if the threshold values are based on packets. But, if we conduct the same experiment with the bandwidth threshold value, the SD-IoT network throughput decreases 40% with four and 80% with eight attack nodes. The CPU utilization increased on experiments conducted with the packet threshold value and remained approximately constant with experiments with 0.5 Kbytes/sec threshold value and also near to constant or fluctuated a little bit with 1 Kbytes/sec threshold values. The attack detection time is high if the experiments are conducted with the packet-based threshold, but with the 0.5 Kbytes/sec bandwidth threshold value, the attack detection time shoots down from 45 to 5 seconds compared to packet-based 140 to 50 second. The experiments conducted with 1 Kbytes/sec threshold value attack detection time were 122 to 65 seconds; all these values were 4 to 8 attack nodes.

The burst mode or packet frequency has much more effect on SD-IoT network throughput with the packet-based threshold experiments. It decreased from 95% to 99%, and in the experiments conducted with the bandwidth, it decreased up to 67%, 73%, 85%, and so we noted that SD-IoT network throughput decreases slowly with increased burst node. Instead, in a packet-based threshold, it shoots down as we double the initial burst frequency. The attack detection time decreases rapidly with the experiments conducted with 0.5 KBytes/sec threshold value up to 40 seconds to 1 second. If we double the initial burst frequency, there is a little bit change in attack detection time, but there is no big difference. The attack detection time decreased with the same ratio of 91 to 5 seconds, with the experiments conducted with 1 Kbytes/sec threshold. Suppose burst frequency is doubled than the initial value, in that case, the attack detection time does not decrease much rapidly with the packet-based threshold value. It decreases to 41, 23, 14, 12, 6 seconds to different experiments with different Burst size.

The experiments are conducted with a packet-based threshold value in the packet payload size parameter; the SD-IoT network throughput values remain constant with different packet payload sizes. But in the experiments conducted with the bandwidth threshold value, it was not stable, and in consecutive two to three experiments, it did not change very much. There is no significant difference in attack detection time with the packet-based threshold value experiment as with all size detection, and time was 28 to 30 seconds. But the experiments with the bandwidth-based threshold value, the attack detection time is decreased as per increment in packet payload. It is also noted that with the high bandwidth threshold value, attack detection time decreased such as 150 to 5 seconds, through different packet payload experiments. With the low bandwidth threshold value, the attack detection time is decreased with low value such as 26 to 1 seconds with varying sizes of packet payload. Meanwhile, we can say that the attack detection time does not change much and remains approximately equal to constant with packet-based threshold value experiments and varies with respective packet payload experiments with the bandwidth threshold value.

We have observed that C-DAD detects DDoS attack with time-varying according to the concerned experiments. In the burstly scenario, the maximum attack detection time is 40 seconds while the minimum time is 6.7 seconds with the Burst mode parameter, whereas, with packet payload parameter, it takes approximately 30 seconds. With attack nodes parameter, it takes the maximum time as 140 seconds and the minimum time is 26 seconds. In experiment 2, it detects in 1 second with burst mode, with attack nodes, the maximum time is 45 seconds and minimum time 5 seconds, and maximum time with packet payload is 26, and minimum time is 1 second. In the third experiment, attack detection time is approximately 1 second with burst mode and maximum 122 seconds and minimum 13 seconds with attack nodes. With packet payload the maximum time is 150 seconds and minimum 1 second. According to the results and discussion, we conclude that the proposed framework with the C-DAD application efficiently detects the DDoS attack.

VI. CONCLUSION

In this article, we presented a novel C-DAD (Counter-based DDoS Attack Detection) framework built on top of the SDNWISE framework, to analyze and detect the DDoS attack with an affordable time. The framework consists of the SDNWISE controller, IoT controller SOPS and IoT devices. The basis of the proposed framework consists of different counter-based functions in its sub-modules, such as Flow Monitor and Flow Analyzer, to dynamically detect DDoS attack in the SD-IoT network.

This article aims to provide the dynamic and programmable DDoS attack detection solution for the SD-IoT network. This article initially discusses the general architecture of IoT, and also describes the SDNWISE framework in detail with its components. Then, we explain the proposed framework and C-DAD algorithms for DDoS attack detection. The algorithm and framework were tested through different experiments. We have extensively analyzed the proposed framework's performance for attack detection time and other parameters such as SD-IoT network throughput, CPU and memory utilization, etc. The proposed framework detects the attack efficiently, hence minimizing the time of attack detection with a tolerable impact on CPU and memory.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[2] R. Taylor, D. Baron, and D. Schmidt, "The world in 2025–predictions for the next ten years," in *Proc. 10th Int. Microsyst., Packag., Assem. Circuits Technol. Conf. (IMPACT)*, Oct. 2015, pp. 192–195.

[3] A. Kanuparthi, R. Karri, and S. Addepalli, "Hardware and embedded security in the context of Internet of Things," in *Proc. ACM Workshop Secur., Privacy Dependability Cyber Vehicles (CyCAR)*, 2013, pp. 61–64.

[4] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: A novel honeypot for revealing current IoT threats," *J. Inf. Process.*, vol. 24, no. 3, pp. 522–533, 2016.
[5] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.
[6] J. Zhang, H. Chen, L. Gong, J. Cao, and Z. Gu, "The current research of IoT security," in *Proc. IEEE 4th Int. Conf. Data Sci. Cyberspace (DSC)*, Jun. 2019, pp. 346–353.
[7] F. S. Dantas Silva, E. Silva, E. P. Neto, M. Lemos, A. J. V. Neto, and F. Esposito, "A taxonomy of DDoS attack mitigation approaches featured by SDN technologies in IoT scenarios," *Sensors*, vol. 20, no. 11, p. 3078, May 2020.
[8] N. Kumar, N. Mittal, P. Thakur, and R. Srivastava, "Analysis of different detection and mitigation algorithm of ddos attack in software-defined Internet of Things framework: A review," in *Recent Trends and Advances in Artificial Intelligence and Internet of Things*. Cham, Switzerland: Springer, 2020, pp. 597–607.
[9] N. M. Abdelazim, S. F. Fahmy, M. A. Sobh, and A. M. B. Eldin, "A hybrid entropy-based DoS attacks detection system for software defined networks (SDN): A proposed trust mechanism," *Egyptian Informat. J.*, May 2020, doi: 10.1016/j.eij.2020.04.005.
[10] Y.-W. Chen, J.-P. Sheu, Y.-C. Kuo, and N. Van Cuong, "Design and implementation of IoT DDoS attacks detection system based on machine learning," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2020, pp. 122–127.
[11] A. M. Zarca, J. B. Bernabe, A. Skarmeta, and J. M. A. Calero, "Virtual IoT HoneyNets to mitigate cyberattacks in SDN/NFV-enabled IoT networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1262–1277, Jun. 2020.
[12] N. Omnes, M. Bouillon, G. Fromentoux, and O. Grand, "A programmable and virtualized network & IT infrastructure for the Internet of Things: How can NFV & SDN help for facing the upcoming challenges," in *Proc. 18th Int. Conf. Intell. Next Gener. Netw.*, 2015, pp. 64–69.
[13] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the Internet-of-Things," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–9.
[14] P. Hu, "A system architecture for software-defined industrial Internet of Things," in *Proc. IEEE Int. Conf. Ubiquitous Wireless Broadband (ICUWB)*, Oct. 2015, pp. 1–5.
[15] Y. Jararweh, M. Al-Ayyoub, A. Darabseh, E. Benkhelifa, M. Vouk, and A. Rindos, "SDIoT: A software defined based Internet of Things framework," *J. Ambient Intell. Humanized Comput.*, vol. 6, no. 4, pp. 453–461, Aug. 2015.
[16] J. Li, E. Altman, and C. Touati, "A general SDN-based IoT framework with NVF implementation," *ZTE Commun.*, vol. 13, no. 3, pp. 42–45, 2015.
[17] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, Apr. 2014.
[18] F. Idris and S. Hameed, "Software defined security service provisioning framework for Internet of Things," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 12, pp. 1–15, 2016.
[19] P. Diogo, L. P. Reis, and N. V. Lopes, "Internet of Things: A system's architecture proposal," in *Proc. 9th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Jun. 2014, pp. 1–6.
[20] S. Chakrabarty and D. W. Engels, "A secure IoT architecture for smart cities," in *Proc. 13th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2016, pp. 812–813.
[21] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 513–521.
[22] H. Mostafaei and M. Menth, "Software-defined wireless sensor networks: A survey," *J. Netw. Comput. Appl.*, vol. 119, pp. 42–56, Oct. 2018.
[23] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet*, vol. 43. Hoboken, NJ, USA: Wiley, 2011.
[24] I. Howitt and J. A. Gutierrez, "IEEE 802.15.4 low rate–wireless personal area network coexistence issues," in *Proc. IEEE Wireless Commun. Netw. (WCNC)*, Mar. 2003, pp. 1481–1486.
[25] C. Vandana, "Security improvement in IoT based on software defined networking (sdn)," *Int. J. Sci., Eng. Technol. Res.*, vol. 5, no. 1, pp. 2327–4662, 2016.
[26] T. Ninikrishna, S. Sarkar, R. Tengshe, M. K. Jha, L. Sharma, V. K. Daliya, and S. K. Routray, "Software defined IoT: Issues and challenges," in *Proc. Int. Conf. Comput. Methodol. Commun. (ICCMC)*, Jul. 2017, pp. 723–726.
[27] D. Yin, L. Zhang, and K. Yang, "A DDoS attack detection and mitigation with software-defined Internet of Things framework," *IEEE Access*, vol. 6, pp. 24694–24705, 2018.
[28] P. Bull, R. Austin, E. Popov, M. Sharma, and R. Watson, "Flow based security for IoT devices using an SDN gateway," in *Proc. IEEE 4th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2016, pp. 157–163.
[29] X. Luo, Q. Yan, M. Wang, and W. Huang, "Using MTD and SDN-based honeypots to defend DDoS attacks in IoT," in *Proc. Comput., Commun. IoT Appl. (ComComAp)*, Oct. 2019, pp. 392–395.
[30] N. Ravi and S. M. Shalinie, "Learning-driven detection and mitigation of DDoS attack in IoT via SDN-cloud architecture," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3559–3570, Apr. 2020.
[31] F. A. Fernandes Silveira, F. Lima-Filho, F. S. D. Silva, A. de Medeiros Brito, Jr., and L. F. Silveira, "Smart detection-IoT: A DDoS sensor system for Internet of Things," in *Proc. Int. Conf. Syst., Signals Image Process. (IWSSIP)*, Jul. 2020, pp. 343–348.
[32] Y. Yang, J. Wang, B. Zhai, and J. Liu, "IoT-based DDoS attack detection and mitigation using the edge of SDN," in *Proc. Int. Symp. Cyberspace Saf. Secur.* Cham, Switzerland: Springer, 2019, pp. 3–17.
[33] A. A. Gendreau and M. Moorman, "Survey of intrusion detection systems towards an end to end secure Internet of Things," in *Proc. IEEE 4th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2016, pp. 84–90.
[34] H. Mustapha and A. M. Alghamdi, "DDoS attacks on the Internet of Things and their prevention methods," in *Proc. 2nd Int. Conf. Future Netw. Distrib. Syst. (ICFNDS)*, 2018, p. 4.
[35] Q. Yan, Q. Gong, and F. R. Yu, "Effective software-defined networking controller scheduling method to mitigate DDoS attacks," *Electron. Lett.*, vol. 53, no. 7, pp. 469–471, Mar. 2017.
[36] W. Xiulei, C. Ming, W. Xianglin, and Z. Guomin, "Defending DDoS attacks in software defined networking based on improved Shiryaev–Roberts detection algorithm," *J. High Speed Netw.*, vol. 21, no. 4, pp. 285–298, Nov. 2015.
[37] L. Wei and C. Fung, "FlowRanger: A request prioritizing algorithm for controller DoS attacks in software defined networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 5254–5259.
[38] J. Cui, M. Wang, Y. Luo, and H. Zhong, "DDoS detection and defense mechanism based on cognitive-inspired computing in SDN," *Future Gener. Comput. Syst.*, vol. 97, pp. 275–283, Aug. 2019.
[39] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2015, pp. 77–81.
[40] A. B. Dehkordi, M. Soltanaghaei, and F. Z. Boroujeni, "The DDoS attacks detection through machine learning and statistical methods in SDN," *J. Supercomput.*, pp. 1–33, Jun. 2020, doi: 10.1007/s11227-020-03323-w.
[41] J. Galeano-Brajones, J. Carmona-Murillo, J. F. Valenzuela-Valdés, and F. Luna-Valero, "Detection and mitigation of DoS and DDoS attacks in IoT-based stateful SDN: An experimental approach," *Sensors*, vol. 20, no. 3, p. 816, Feb. 2020.
[42] R. M. A. Ujjan, Z. Pervez, K. Dahal, A. K. Bashir, R. Mumtaz, and J. González, "Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN," *Future Gener. Comput. Syst.*, vol. 111, pp. 763–779, Oct. 2020.
[43] R. B. Shohani and S. A. Mostafavi, "Introducing a new linear regression based method for early DDoS attack detection in SDN," in *Proc. 6th Int. Conf. Web Res. (ICWR)*, Apr. 2020, pp. 126–132.
[44] A. Wani and S. Revathi, "DDoS detection and alleviation in IoT using SDN (SDIoT-DDoS-DA)," *J. Inst. Eng. B*, vol. 101, no. 2, pp. 117–128, Apr. 2020.
[45] SDN-WISE. *SDN-Wise the Stateful Software Defined Networking Solution for the Internet of Things*. Accessed: Mar. 2020. [Online]. Available: https://sdnwiselab.github.io/
[46] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Proc. 31st IEEE Conf. Local Comput. Netw.*, Nov. 2006, pp. 641–648.
[47] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki—A lightweight and flexible operating system for tiny networked sensors," in *Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, Nov. 2004, pp. 455–462.

JALAL BHAYO was born in Sanghar, Pakistan. He received the B.Sc. degree in computer science from the University of Sindh, Jamshoro, and the M.S. degree in computer science from the National University of Computer and Emerging Science (NUCES), Karachi, where he is currently pursuing the Ph.D. degree in computer science. He is currently with the IT Security Laboratory, NUECS. He is also a Lecturer in computer science with the Government of Sindh. He has industrial expertise in different networking related products and also received Cisco certification. His research interest areas include the Internet of things, network, Web security, and SDN with respect to security.

SUFIAN HAMEED received the Ph.D. degree in the field of networks and information security from the University of Göttingen, Germany. He is currently an Associate Professor with the Department of Computer Science, National University of Computer and Emerging Sciences, Pakistan, where he also leads the IT Security Laboratory. The research laboratory studies and teaches security problems and solutions for different types of information and communication paradigms. His research interests include network security, Web security, mobile security and secure architectures, and protocols for cloud and the IoTs.

SYED ATTIQUE SHAH received the Ph.D. degree from the Institute of Informatics, Istanbul Technical University, Istanbul, Turkey. During his Ph.D., he was a Visiting Scholar with National Chiao Tung University, Taiwan, The University of Tokyo, Japan, and the Tallinn University of Technology, Estonia, where he completed the major content of his thesis. He was an Assistant Professor and the Chairperson with the Department of Computer Science, BUITEMS, Quetta, Pakistan. He is currently a Lecturer with the Data Systems Group, Institute of Computer Science, University of Tartu, Estonia. His research interests include big data analytics, cloud computing, information management, and the Internet of Things.