IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 12, 2019, accepted July 27, 2019, date of publication August 1, 2019, date of current version August 19, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2932438

SeArch: A Collaborative and Intelligent NIDS Architecture for SDN-Based Cloud IoT Networks

TRI GIA NGUYEN[1], (Member, IEEE), TRUNG V. PHAN[2], (Member, IEEE), BINH T. NGUYEN[3,4], CHAKCHAI SO-IN[5], (Senior Member, IEEE), ZUBAIR AHMED BAIG[6], AND SURASAK SANGUANPONG[7], (Member, IEEE)

[1]Faculty of Information Technology, Duy Tan University, Da Nang 50206, Vietnam
[2]Chair of Communication Networks, Technische Universität Chemnitz, 09126 Chemnitz, Germany
[3]Sector of International Education, Hanoi University of Science and Technology, Hanoi 11615, Vietnam
[4]Information Technology Services, Victoria University of Wellington (VUW-IT), Wellington, New Zealand
[5]Applied Network Technology Laboratory, Department of Computer Science, Faculty of Science, Khon Kaen University, Khon Kaen 40002, Thailand
[6]School of Information Technology, Deakin University, Geelong, VIC 3220, Australia
[7]Applied Network Research Laboratory, Department of Computer Engineering, Faculty of Engineering, Kasetsart University, Bangkok 10900, Thailand

Corresponding author: Chakchai So-In (chakso@kku.ac.th)

This work was supported in part by the Enthuse Company Ltd. under Grant Ent-KKU-2560-01, in part by the Khon Kaen University Grant, in part by the Kasetsart University Grant, and in part by the Thailand Research Fund (TRF) through the International Research Network Program under Grant IRN61W0006.

ABSTRACT The explosive rise of intelligent devices with ubiquitous connectivity have dramatically increased Internet of Things (IoT) traffic in the cloud environment and created potential attack surfaces for cyber-attacks. Traditional security approaches are insufficient and inefficient to address security threats in cloud-based IoT networks. In this vein, software defined networking (SDN), network function virtualization (NFV), and machine learning techniques introduce numerous advantages that can effectively resolve cybersecurity matters for cloud-based IoT systems. In this paper, we propose a collaborative and intelligent network-based intrusion detection system (NIDS) architecture, namely *SeArch* for SDN-based cloud IoT networks. It composes a hierarchical layer of intelligent IDS nodes working in collaboration to detect anomalies and formulate policy into the SDN-based IoT gateway devices to stop malicious traffic as fast as possible. We first describe a new NIDS architecture with a comprehensive analysis in terms of the system resource and path selection optimizations. Next, the system process logic is extensively investigated through main consecutive procedures, including initialization, runtime operation, and database update. Afterward, we conduct a detailed implementation of the proposed solution in an SDN-based environment and perform a variety of experiments. Finally, evaluation results of the *SeArch* architecture yield outstanding performance in anomaly detection and mitigation as well as bottleneck problem handling in the SDN-based cloud IoT networks in comparison with existing solutions.

INDEX TERMS Internet of Things security, software defined networking, network function virtualization, machine learning, intrusion detection system, distributed cloud computing.

I. INTRODUCTION

The advancement of Internet of Things (IoT) has been bringing enormous capabilities for ubiquitously intelligent connectivity and applications in many domains of human life [1], [2]. Smarter devices can provide a smart and active life for human by enabling sensing and actuation abilities, contextual awareness [3], [4]. Recently, the IoT appliances have been exponentially increased due to a wide range of new technologies [1] such as sensors, wireless communications and cloud computing technologies, e.g., Software-Defined Networking (SDN) and Network Function Virtualization (NFV) [5]. A good illustration, Cisco Systems [6], forecasts the global mobile data traffic projections and growth trends for a period of time from 2017 to 2022, in which there will be 12.3 billion mobile-connected devices by 2022, and the global mobile data traffic will reach 77 exabytes every

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

month by 2022. The tremendous amount of data would be absorbed into the Internet consisting of smart-home devices, autonomous vehicles, wearable devices, environmental sensors, and almost anything we can imagine. Consequently, the opportunities from the development of IoTs are endless, and its capabilities and potential will be tangible very soon as a vast number of IoT devices are getting connected to the Internet day by day. On the other hand, IoT network systems present new potential cyber-attack surfaces for malicious attackers leading to tremendous economic and reputation destruction for system operators/providers [7], [8], if there are no correctly protection solutions.

Fortunately, the network softwarization including SDN and NFV cloud technologies are representing a major breakthrough in Telco industries, by providing several benefits regarding dynamics, flexibility, and manageability. Concerning network security, these two key enablers of cloud computing technologies are obtaining a great momentum by introducing dynamic and flexible security protection mechanisms to cloud environment [9]. Although, a variety of studies based on SDN/NFV technologies have been proposed to better cope with IoT security threats [10]–[16]. However, current solutions still face with some critical problems such as bottleneck issues [11]–[15] and lacking of collaboration [16], [17] while providing security services or mechanisms for cloud-based IoT networks. In addition, due to the huge quantity of IoT devices, it is always challenging for every network operator to create an effective defense mechanism against cyber attacks in IoT networks [7].

Therefore, in this article, we propose a novel collaborative and intelligent network-based intrusion detection system (NIDS) architecture to effectively defense against network-related cyber attacks in SDN-based cloud IoT networks, entitled *SeArch*. This security architecture consists of a hierarchical distribution of NIDS nodes, including Edge-IDS, Fog-IDS, and Cloud-IDS, respectively. These IDSs are based on machine learning/deep learning algorithms for their detection operations, and those located in the same computing layer can be in a distributed design. In particular, Edge-IDS is a lightweight security application integrated into an SDN-based IoT gateway in the edge computing level, Fog-IDS located in the fog computing layer runs as an SDN application on top of SDN controller, and Cloud-IDS is an IoT security application running on the cloud computing level with enough computation power and storage resources. This architecture introduces an effective collaboration way among IDS nodes in network-related anomaly IoT traffic detection by setting up communication channels among nodes for data synchronization and load balancing.

Our significant contributions can be listed as follows:

- Firstly, we analyze existing security solutions and provide motivations for applying machine learning/deep learning-based detection techniques to cyber attacks in cloud-based IoT networks. Afterward, we show the resource consumption problem at the edge computing level by our prior experiment.
- Secondly, we propose a new security architecture, *SeArch*, representing a collaborative and intelligent NIDS framework in SDN-based cloud IoT networks, in which an arrangement of three layers of IDS nodes, i.e., Edge-IDS, Fog-IDS, and Cloud-IDS, is introduced with an effective collaboration among nodes.
- Furthermore, we conduct a thorough analysis of the system and take the resource management and the overhead of communication of the proposed solution into account. Then we formulate a novel system resource optimization and optimal path selection scheme.
- Finally, we carry out comprehensive experiments in an SDN-based cloud IoT emulation network. An extensive comparison of *SeArch* with existing solutions shows significant improvements in anomaly detection and mitigation as well as performance bottleneck handling.

The rest of this article is constructed as follows. Section II shows background knowledge about the SDN-based cloud IoT networks and network-related security threats. Next, research motivations will be provided in section III. Details of our proposed *SeArch* architecture are presented in section IV. Section V presents our deployment example and experiments, and result analysis will be provided in section VI. Finally, section VII concludes our study and draws some future developments.

II. PRELIMINARIES

A. SDN-BASED CLOUD IoT NETWORKS

As illustrated in Figure 1, the modern cloud-based IoT networks is often divided into three primary levels: Edge computing, Fog computing, and Cloud computing [1], [11], [18], [19] which correspond to three layers of IoT system: Perception, Distribution Network, and Application. A brief description of three computing levels is given as follows.

1) EDGE COMPUTING LEVEL

The edge computing level mainly covers edge nodes (e.g., routers switches and small/macro base stations) powered by mobile edge computing (MEC) technology [20]. However, in the SDN-based cloud vein, we regularly consider SDN-based IoT gateways which support protocols (e.g., OpenFlow and NetConf) and connect to the SDN control plane located in the fog computing level. An SDN-based IoT gateway connects to IoT devices via several protocols [1] such as ZigBee, WirelessHART, RUBEE, WiFi, Ethernet, and second-generation 2G/3G/4G/5G. In addition, the edge computing level is located at the distribution network layer of the IoT systems which can provide real-time connection, services, and security for limited resource capacity IoT devices in the perception layer [19] by leveraging the capability of MEC technology. Besides, it is noted that computational capability at the edge devices is one of the significant challenges for the edge computing level because of the computation offloading [21], [22]. For example, massive IoT devices may require for different compute-intensive

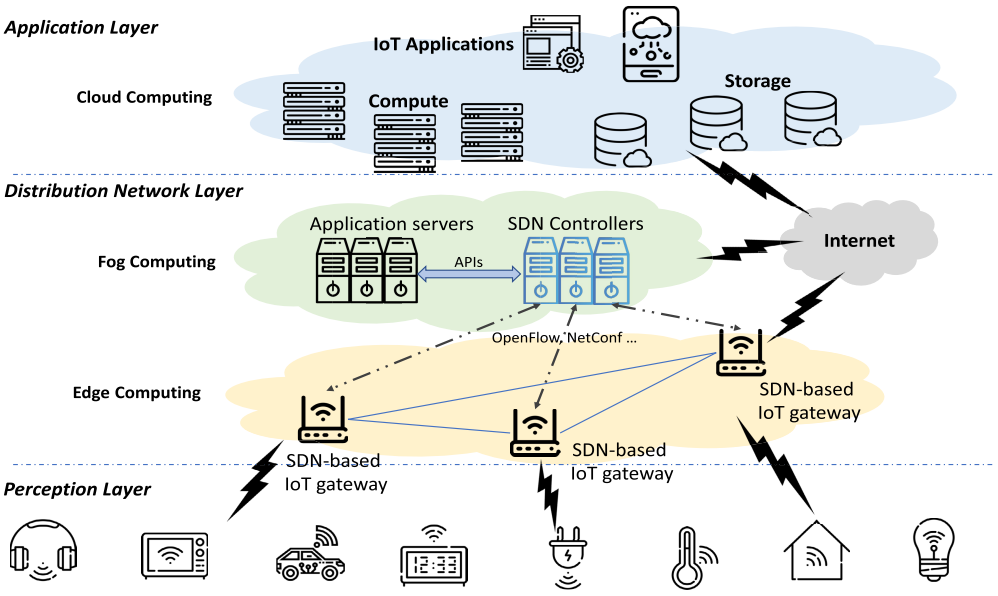

FIGURE 1. Overview of SDN-based Cloud IoT Networks.

services/applications from a mobile edge station, which can result in an outage situation.

2) FOG COMPUTING LEVEL

The fog computing level is also placed at the distribution network layer. It mainly consists of SDN controllers and SDN application servers, in which they could be formed as a distributed manner. These SDN controllers communicate with SDN-based IoT gateways via southbound protocols (e.g., OpenFlow and NetConf); meanwhile, northbound APIs are used for data exchange with application servers and the cloud computing level. Hence, the fog computing can provide not only enough computational resources but also low latency and compute-intensive applications [12], [19], which makes it become a great place to deploy IoT security applications.

3) CLOUD COMPUTING LEVEL

A standard definition for the cloud computing level is given by the National Institute for Standards and Technology (NIST) in 2011 [23], it reports "*a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*". Accordingly, this computing level is majorly responsible for storage, processing and accessing of data produced by a vast number of IoT devices. In other words, high computational applications and big data storage should be placed at the cloud computing level [19].

B. NETWORK SECURITY THREATS

Cloud environment for IoT networks and platforms provides not only connectivity among IoT devices and applications but also distributed computational resources and storage. Consequently, several potential security vulnerabilities have been exploited recently by attackers [10]. As illustrated in Figure 2, we can categorize common cyber-attacks into Network-related and other groups which could happen and seriously harm the cloud-based IoT networks. However, in this research, we only pay attention to well-known network-related security threats which are briefly described as follows.

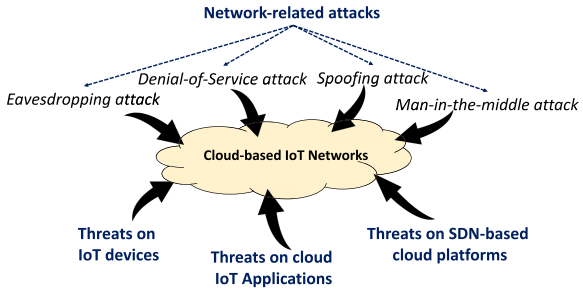

FIGURE 2. Common security threats in the cloud-based IoT networks.

1) EAVESDROPPING ATTACK

Known as data sniffing, eavesdropping technique [24] is a serious cyber-attack conducted by listening to IoT device communications. In particular, if transferring data are unencrypted in an insecure channel, sniffers can extract sensitive information from those communications such as device credentials or configurations. Eavesdropping attacks are challenging to detect and prevent completely; this is because adversaries do not cause network transmissions to appear to be abnormal.

2) DENIAL-OF-SERVICE ATTACK

Denial-of-Service (DoS) or Distributed DoS (DDoS) attacks [8] are the most common and dangerous cyber-threats in cloud-based IoT environment. With the purpose of flooding network links and IoT devices with an enormous traffic volume, adversaries can quickly exhaust network and computational resources resulting in the unavailability of the IoT communication system. Different techniques can be used to launch a saturation attack, such as ICMP flood, TCP/UDP SYN flood, TearDrop, and Low & Slow DDoS. For example, Mirai botnet network is built based on a swarm of more than 400,000 vendor/technology-specific IoT devices that saturated a French WebHost in September 2016 [8] with 1 Tbps of DDoS traffic.

3) SPOOFING ATTACK

The adversarial purpose of a spoofing attack is to send malicious traffic to destination IoT networks and devices but seem legitimate traffic patterns. For instance, attackers can launch an eavesdropping attack to gather information about authorized accesses, then spoof attack traffic with legitimate information such as IP addresses, hence gaining access to IoT network system [25].

4) MAN-IN-THE-MIDDLE ATTACK

A man-in-the-middle attack (MITM) is an advanced version of the spoofing attack where an adversary lies on the network path between two IoT devices in communication. The attacker impersonates both devices and relays traffic by independently communicating with each endpoint in order to intercept the transferring data between victims. This technique makes two parties believe in their communication channel without any doubts about delaying, cloning, replaying, spoofing or dropping packets. Accordingly, the MITM attack can bring savage effects to IoT systems in case of sensitive information captured by an adversary such as control traffic or key exchange data, hence causing insurmountable security problems for IoT system [26].

III. RESEARCH MOTIVATIONS

A. EXISTING SOLUTIONS FOR SECURING CLOUD-BASED IoT NETWORKS

Many research efforts [10], [27] have been proposed to secure cloud-based IoT networks. A recent research [11] introduces a multi-level DDoS mitigation framework to defend against DDoS attacks for industrial IoT networks. This mechanism leverages SDN to manage a large number of industrial IoT devices and to mitigate DDoS attack traffic. However, this solution is vulnerable to bottleneck problems because the detection engine is centrally placed on top of the SDN controller. Moreover, high latency between detection engine and the data plane would reduce the attack mitigation performance. Similarly, a fog-assisted intrusion detection/prevention system [12] and an artificial intelligence-based two-stage intrusion detection system [13] are proposed to deal with attacks in IoT networks relying on SDN and cloud platform technologies. As a result, these mechanisms can be collapsed because of a centralized control in case of a high data processing load.

Authors in [17] present a distributed mechanism for intrusion detection system utilizing SDN and programmable forwarding devices (e.g., OpenvSwitch), in which intrusion detection system is located at the forwarding devices, and it is running as a security service in the data plane. One of the main drawbacks of this mechanism is lacking collaboration among detection engines; hence if a new kind of attacks presents, it takes a long time to detect or even cannot recognize the new attack due to lacking updates from other nodes in the system. A study in [16] shows the same approach for placing the detection engine in the SDN-based IoT gateway as an extra security function. Accordingly, these two methods [16], [17] are still not capable of handling with different and new cyber-attacks in IoT networks. Another research proposes a novel framework [14], called ATLANTIC, for anomaly traffic detection, classification, and mitigation based on the capability of SDN. ATLANTIC calculates deviations from collected information in flow tables in the data plane and then uses machine learning algorithms to classify traffic flows. Again, this framework does not mention or resolve the centralized bottleneck problem, but it only focuses on intelligently recognizing new attacks.

Additionally, in [28], an SDN-IoT architecture based on Network Function Virtualization is proposed with virtual IoT gateway which brings dynamics, scalability, and elasticity to the control and management of IoT network traffic in the data plane but does not open considerable and detailed discussions regarding security issues. Another study [15] presents a framework to overcome the big data problem by analyzing IoT traffic patterns in lower layers instead of evaluating the values in the application layer. In which, the volume of data is significantly reduced before coming to Internet or other places; however, it places very high pressures on devices in the data plane, and it is vulnerable to bottleneck issues in case of a high traffic load. Authors in [29] introduce a new host-based intrusion detection and mitigation architecture based on SDN and OpenFlow protocol to protect smart IoT devices at network-level in home environment. Nonetheless, this framework is insecure to bottleneck and scalability problems because of its centralized design, and it requires high computational efforts for traffic monitoring in realtime. Consequently, these mentioned approaches are only applicable for addressing some specific security problems and can operate efficiently under certain conditions.

To sum up, previous research are still facing with critical problems, i.e., bottleneck issues [11]–[15] and lacking of collaboration [16], [17], in order to better secure cloud-based IoT networks. In this article, therefore, this motivates us to come up with our new security architecture for efficiently securing SDN-based cloud IoT networks, which can benefit the advantages of existing solutions while avoiding their drawbacks.

B. INTELLIGENCE-BASED APPROACH FOR SECURITY IN IoT NETWORKS

1) MACHINE LEARNING TECHNIQUES FOR CYBER SECURITY

Machine learning techniques have been exhibiting notable successes in classification problems in many computer networks-related areas [17], [30], [31], recently. Because it provides a general solution to solve complex classification problems where a phenomenon model is hugely complex to derive or very dynamic to be assumed in mathematics, which makes the machine learning's popularity nowadays. Concerning the classification of machine learning techniques, although there are many ways to classify them, we consider a common classification based on learning methods [30], [31] including *reinforcement*, *supervised*, and *unsupervised* in this work. Most research applying reinforcement learning approach are for optimization problems or finding an optimal solution for a specific situation [32]. Meanwhile, supervised and unsupervised learning algorithms are mostly for classifying and clustering input data into separated groups. Therefore, two latter are seen in many studies related to security solutions in IoT networks [27], which motivates us to take a machine learning-based approach into account with the purpose of intelligently classifying IoT network traffic in this research.

To make the above motivation clear, we conduct a prior experiment on three machine learning-based security solutions for a simple SDN-based network in MaxiNet emulator [33]. The emulated network comprises a Web server and 08 hosts (04 attackers and 04 benign users); they are all based on Linux containers and connect to an OpenFlow switch (OpenvSwitch). We leverage BoNeSi tool [34] to launch DDoS TCP SYN attacks with 100, 200, and 300 Mbps from attackers to the Web server in the SDN network, then precision and accuracy metrics are recorded for the evaluation of this attack detection performance. Those security solutions based on a lightweight traffic classifier [35] (Support Vector Machine - SVM), a moderate traffic classifier [36] (Self Organizing Map - SOM), and an intensive and complex traffic classifier [37] (Stacked Autoencoder Deep Learning approach - SAE), respectively. Note that, for training phase, we use public data sets, CAIDA [38], [39], to extract training samples[1] for detection engines. As shown in Table 1, the DDoS detection performance[2] of three security approaches are remarkable with a trained classification algorithm. Moreover, it shows that a more complex classification algorithm gives a higher chance to recognize the DDoS attack presence.

2) COMPUTATION ISSUES AT THE EDGE COMPUTING LEVEL IN SDN-BASED CLOUD IoT NETWORKS

As discussed in Section II-A, the computational capacity is one of the significant issues for SDN-based IoT gateway devices because the edge device is a perfect location for the deployment of applications [11], [19] such as access control, intrusion detection, and data encryption. For instance, a security application which requires real-time and complex processes, it always needs for an intensive computational resource from the gateway device. As a result, the edge device is unable to perform all processing tasks well and may become overloaded. To clarify this issue, we again utilize the emulation setup in the above section and extract the CPU utilization[3] of the machine hosted three security applications based on SVM, SOM, and SAE algorithms during attacks. As can be seen in Figure 3, the results show that the required computational capability is proportional to the complexity of machine learning or deep learning algorithms, and there are significant differences among cases. Hence, the placement of security applications plays a crucial role in the operating performance of the edge devices.

From above analyses, accordingly, in order to eliminate this problem at the edge device while providing efficient security services for SDN-based cloud IoT networks, we propose a collaborative and intelligent network-based intrusion detection system in next section.

TABLE 1. Anomaly detection performance of security applications during different DDoS attack volumes.

Algorithms	SVM	SOM	SAE
Precision results (%)			
100 Mbps	94.34	95.40	97.80
200 Mbps	93.06	96.63	97.63
300 Mbps	93.23	96.54	97.65
Accuracy results (%)			
100 Mbps	94.56	96.85	97.98
200 Mbps	94.23	96.67	97.67
300 Mbps	94.12	96.78	97.90

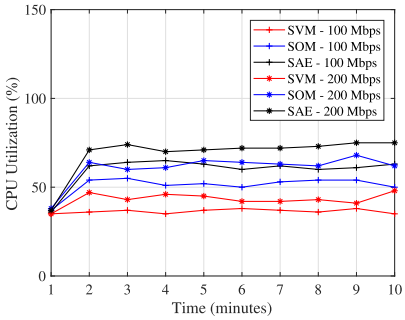

FIGURE 3. CPU utilization of a server running security applications during different DDoS attack volumes.

IV. COLLABORATIVE AND INTELLIGENT NIDS FOR SDN-BASED CLOUD IoT NETWORKS

In this section, we elaborate the proposed *SeArch* architecture for network anomaly detection system in SDN-based cloud IoT networks. We firstly describe the overall architecture and

[1] Based on studies in [17], [40], we train SVM and SOM classifiers using simple features consisting of number of flows, number of packet per flow, number of byte per flow, flow duration, growth of client ports, and protocol. For the deep learning-based SAE classifier, we apply a set of 9 features for TCP traffic and a set of 6 features for ICMP traffic as the same in [37].

[2] Traffic is classified into two classes, including normal and attack.

[3] We observe the CPU usage of the core running the security application. The machine is Intel Core i7-4790 (4 cores) 3.60 GHz and 16 GB DDR3 1600 MHz.

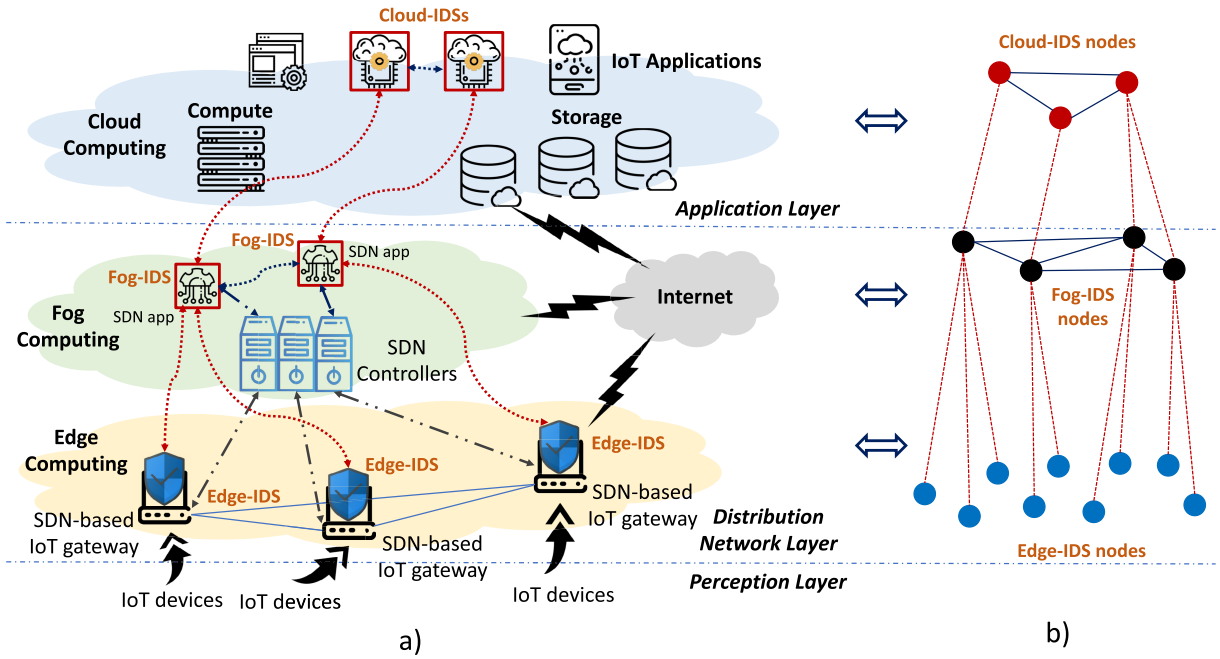

FIGURE 4. a) Proposed collaborative and intelligent NIDS architecture for SDN-based cloud IoT Networks. b) NIDS node graph of the proposed architecture.

components, then operation workflow and proposed algorithms will be provided hereinafter.

A. OVERALL ARCHITECTURE

In order to minimize mentioned issues, i.e., performance bottleneck, lacking collaboration, and resource overconsumption at the edge computing (see section III), we propose a collaborative and intelligent NIDS architecture in SDN-based cloud IoT networks that is illustrated in Figure 4. The introduced security architecture consists of three main layers of IDS including Edge-IDS, Fog-IDS, and Cloud-IDS, respectively. In which, these IDSs are both placed in hierarchical and vertical dimensions. However, IDSs, which are in the same computing layer, can be in a distributed form. In particular, Edge-IDS is a security application integrated into an SDN-based IoT gateway device, Fog-IDS is installed as an SDN application above SDN controllers, and Cloud-IDS is an IoT security application running on the cloud with enough computation and storage resources.

Regarding the collaboration among IDSs, each Edge-IDS is designed to have an online communication channel[4] to a Fog-IDS for information/data exchange. Likewise, Fog-IDS and Cloud-IDS establish their channels to send and receive data. In addition, Fog-to-Fog-IDS and Cloud-to-Cloud-IDS communications are set up for data/information sharing and updating, and there are no channels between Edge-IDSs. The main objective behind this arrangement is to reduce information exchange latency and to avoid outage problems for the edge devices because the higher layers should mainly conduct data/information exchange while the edge devices should focus on processing IoT traffic.

B. MULTI-LEVEL NIDS IN SDN-BASED CLOUD

From the above arrangement, we now expose each computing level IDS in terms of the placement, functionality and degree of collaboration among others in details.

Edge-IDS: This element is placed at the edge computing level and runs as an intrusion detection engine which resides in an SDN-based IoT gateway [11] powered by MEC technology [20]. The main tasks of an Edge-IDS are to periodically collect and extract traffic flow statistics information from a local IoT network to derive a set of desired features which is then fed into a machine learning-based engine. Afterward, anomaly detection and policy-making processes are conducted locally to stop malicious traffic flows in the connected SDN-based IoT gateway device. From our analyses in section III-B.2, therefore, we recommend to utilize only *lightweight* or less time-complexity machine learning-based detection algorithms, i.e., Support Vector Machine [35], as the core of detection engine. Due to the placement of this security mechanism, this arrangement is considered as a *source-based* detection approach. Moreover, for unknown traffic patterns or a high volume incoming traffic, e.g., DDoS attack traffic, the Edge-IDS actively forwards data to the upper layer (Fog-IDS) for further processes via the online channel, and we will elaborate further later on.

[4]Each IDS at a layer is configured to keep in touch with its collaborators using sockets referred to as the online channels.

Fog-IDS: At the fog computing level, the network-based detection engine operates as SDN applications in a distributed manner with data synchronization/updates among others. Because of the sufficient power supply, these Fog-IDSs can perform machine learning-based detection algorithms with *moderate* complexity and resource consumption, e.g., Self Organizing Map - SOM. By doing so, the Fog-IDS is expected to provide low latency and compute-intensive security analyses in comparison with the Edge-IDS. Furthermore, a Fog-IDS should act as an upper or aggregation engine of some Edge-IDSs from the lower layer in order to process and categorize data (unknown and load-balancing). In brief, the main tasks of a Fog-IDS are to aggregate and extract received data to derive a set of desired features, and then feed these features into a machine learning-based engine for its anomaly detection. Subsequently, it makes policies and applies to the dedicated SDN-based IoT gateway if possible; otherwise, it sends to a Cloud-IDS for more advanced analyses regarding unrecognized traffic patterns. Besides, a Fog-IDS can load balance with its Fog-IDS neighbours in case of an outage situation caused by a large amount of data forwarded from Edge-IDS nodes, e.g., a DDoS attack, which is explained in section IV-D.

Cloud-IDS: A Cloud-IDS is referred to a cloud-based detection engine running as IoT applications at the cloud computing layer with the inexhaustible resource supply, and it could be formed in a distributed style. The core engine of a Cloud-IDS can operate machine learning-based detection algorithms with *much higher* complexity and resource consumption, e.g., Deep learning. Therefore the outcome of detection is believed as the most conscious of a traffic pattern sent from a Fog-IDS. Similar to the operation of the Fog-IDS node, this IDS level deals with unhandled data and then sends back appropriate policies to Fog-IDS for immediately stopping malicious traffic flows generated from IoT networks.

It is noted that NIDS engines in our introduced security mechanism are initiated easily as virtual machines with built-images in physical servers powered by the capability and flexibility of SDN/NFV and MEC technologies.

C. SYSTEM MODEL ANALYSIS

Since the proposed security architecture operates as a distributed system in the edge, fog, and cloud computing layers, resource management must be considered. Otherwise, information/data processing may put pressure on some specific nodes in case of high traffic load (e.g., DDoS attack traffic) generated from IoT networks; hence, it can easily lead to a bottleneck problem if no load balancing decision is made in time. Furthermore, data flows among IDSs are also forwarded using the same SDN-based cloud infrastructure, i.e., flow-tables in SDN switches or SDN-based IoT gateways. Accordingly, the overhead of communication of the proposed solution must be appropriately minimized. To achieve these goals, we formulate a novel system resource optimization and optimal path selection scheme thereinafter.

TABLE 2. Notations.

Resource Optimization Formulation	
N	Set of IDS nodes in the edge, fog, and cloud levels
R_i	Resources of an IDS node i
R_i^{max}	Maximum capacity of resource R_i in node i
B_i^{max}	Maximum bandwidth can be processed by node i
B_{ji}	Transferred bandwidth from node j to i, and $B_{ji} \neq B_{ij}$
B_i	Sum of incoming bandwidth to node i s.t. $B_i = \sum_j B_{ji}$ and $B_i \leq B_i^{max}$
$\mathscr{L}_{ji}$	Link capacity between node j and i, and $\mathscr{L}_{ji} = \mathscr{L}_{ij}$
D	An IDS node, and $D \in N$
$D_i^R(B_i)$	Usage of resource R when IDS node i processes bandwidth B_i
Optimal Path Selection	
a_{ij}	The amount data sent from node i to node j
s_i	The available supply at node i
c_{ij}	The cost for flow along the arc between node i and node j

1) EFFICIENT SYSTEM RESOURCE MANAGEMENT

Authors in [41] report the CPU usage of virtual machines is proportional to the amount of processing traffic by measuring the CPU utilization per network I/O rate on a Xen hypervisor. Consequently, a virtual machine consumes more resources when processes more incoming traffic. Therefore, we can approximate how much resources are needed and how many traffic information can be processed by remaining resources. Besides, the amount of incoming traffic is represented by bandwidth. Thus, we design an optimization formula that can calculate optimal bandwidth distribution for each IDS node in order to avoid the exceeded resource capacity.

Firstly, we define some notations in Table 2. As can be seen from Figure 4 b), the proposed mechanism includes a set of N nodes and can distribute traffic based on the concurrent resource usage of each IDS node. A resource R_i of an IDS node i is a type of virtual resources provided by physical servers, $R_i \in \{vCPU, vMemory\}$ in case CPU and memory of the IDS node are considered. Since physical resource has a limit; thus, the value of resource R_i should not exceed its capacity R_i^{max}. A bandwidth B_{ji} denotes the transferred bandwidth from node j to i; therefore, a total bandwidth between two nodes j and i becomes B_{ji}+B_{ij}, which should not be excessive compared to a full duplex link capacity $\mathcal{L}_{ji}$ between them. $D_i^{R_i}(B_i)$ refers the usage of resource R when IDS node i processes bandwidth B_i (Mbps), where $B_i \leq B_i^{max}$. For instance, an IDS node i handles with 10Mbps traffic, then CPU usage of the IDS can be formed as $D_i^{vCPU}(10)$.

From above description of variables, we can set up an optimization formula as follows:

$$maximize \sum_j B_{ji} \tag{1}$$

$$subject\ to\ 0 \leq B_i \leq B_i^{max} \tag{2}$$

$$\forall j, i \in N : B_{ji} + B_{ij} \leq \mathcal{L}_{ji} \tag{3}$$

$$\forall R_i \in \{vCPU, vMemory\}, i \in N : D_i^{R_i}(B_i) \leq R_i^{max}. \tag{4}$$

The main objective of this formulation is calculating maximum incoming traffic on an IDS node i to make a load balancing decision in time avoiding the overload case, and it is written as equation (1). Afterward, we have to determine

conditions to find out a feasible solution for the formula. Equation (2) shows the amount of processing bandwidth B_i, and it must be a positive value and less than the maximum allocated value. Next, equation (3) should be satisfied because the bandwidth of data traffic exchanging between two nodes j and i must not be over a link capacity $\mathcal{L}_{ji}$. Similarly, the resources of a node should not be over its capacity; hence, equation (4) also has to be satisfied.

In this work, we introduce to efficiently manage the system resource by leveraging the capabilities of SDN/NFV technology to initiate IDS nodes as virtual machines over the cloud environment. As a result, the cloud administrator can rely on our proposed calculation in order to set up the right configuration for IDS nodes such as *vCPU, vMemory,* B^{max}, and location. In addition, we now denote this calculation as $ESRM\_func(N)$ to refer to the proposed efficient system resource management in this study.

2) OPTIMAL PATH SELECTION FOR DATA EXCHANGE

In our proposed security architecture, the system is characterized using a graph structure representing nodes and communication channels among them, which is shown in Figure 4 b). In order to meet the mentioned requirement regarding the overhead of communication, we have to find an optimal path, which has the minimum communication cost, between two any IDS nodes in the system. In other words, we are interested in minimizing the cost of transferring data in the proposed system. To achieve this aim, we follow the minimum cost flow formulation in transportation and network flow problems [42]. Firstly, we define some variables as follows: a_{ij} denotes the amount data that must be sent from node i to node j, i.e., unhandled data and load balancing data; s_i represents the available supply of data at a node i, if $s_i \leq 0$; thus, there is a required demand of data at node i; and the shipping cost for flow along the arc between node i and node j is denoted as c_{ij}. In common sense, it is reasonable to assume the system is balanced, i.e., total supply equals total demand, then $\sum_{i=1}^{n} s_i = 0$, where n is the total number of nodes in the system. Accordingly, the minimum cost of the system communication problem can be expressed as follows:

$$minimize \quad \sum_{i,j=1}^{n} c_{ij} a_{ij} \tag{5}$$

$$subjectto \quad \sum_{j=1}^{n} a_{ij} - \sum_{k=1}^{n} a_{ki} = s_i \quad \text{for } i = 1, 2, \ldots, n \tag{6}$$

$$a_{ij} \geq 0 \quad \text{for } i = 1, 2, \ldots, n. \tag{7}$$

From equation (5) and its conditions, we can find the optimum path between two any nodes in the system. The solution for solving this equation is well described in [42]. Therefore, we utilize the solution and the result as a primitive to find paths for node communication in our proposed architecture. For ease of use, we denote equation (5) as a function $FSP\_func(i, j)$ to find the optimum path between node i and j from now on.

D. SYSTEM PROCESS LOGIC

To present our novel *SeArch* security architecture in dealing with network-related cyber-attacks in SDN-based cloud IoT networks, the overall system process logic is provided by three main following consecutive procedures.

1) INITIALIZATION PHASE

Under the perspective of the cloud administrator, we need to investigate the cloud infrastructure of three distributed computing levels properly. Next, we formulate a detailed plan for deployment based on the capabilities of SDN/NFV and MEC technologies. This step focuses on some key points such as the *resource allocation* and the *deployment location* of IDS nodes, and these processes can clearly benefit from our proposals mentioned in sections IV-C.1 and IV-C.2 by using $ESRM\_func(N)$ and $FSP\_func(i, j)$. Additionally, we extensively discussed the placement of machine learning-based detection engine in the SDN-based cloud for IoT networks in sections III-B and IV-B. Then, we use some ready-made data sets [38], [39], [43], [44] and extract samples for training phases of *lightweight*, *moderate*, and *compute-intensive* classification algorithms.

2) RUNTIME OPERATION PHASE

In the next step, we describe how IoT traffic can be classified, and data exchange are done in the *SeArch* framework after the Initialization phase is done. At first, every Edge-IDS

Algorithm 1 Runtime Operation at Edge-IDS Nodes

E_{IDS} ← Set of IDS nodes at the edge computing level and $E_{IDS} \in N$.
B_i ← Amount of total incoming bandwidth of an IDS node i at a certain time.
for $i = 1$ to E_{IDS} **do**
  Collect IoT traffic flow statistics data from gateway.
  Calculate B_i, $D_i^{vCPU}(B_i)$ and $D_i^{vMemory}(B_i)$.
  if $(D_i^{vCPU}(B_i) \leq vCPU_i^{max})$ *and* $(D_i^{vMemory}(B_i) \leq vMemory_i^{max})$ **then**
    continue
  else
    Forward ΔB_i to a designated Fog-IDS, where $\Delta B_i = B_i - B_i^{max}$.
  end if
  Extract *features* from remaining data amount.
  Feed inputs into a *lightweight* IDS.
  Get IDS's *outcome*.
  if *outcome* = *normal* **then**
    continue
  else if *outcome* = *malicious* **then**
    Call **Policy creation and enforcement**.
  else if *outcome* = *unknown* **then**
    Forward to a connected Fog-IDS node.
  end if
end for

Algorithm 2 Runtime Operation at Fog-IDS and Cloud-IDS Nodes

1: $\{F_{IDS}, C_{IDS}\}$ ← Set of IDS nodes at the fog and cloud computing levels and $F_{IDS}, C_{IDS} \in N$.
2: B_i ← Amount of total incoming bandwidth of an IDS node i at a certain time.
3: **for** $i = 1$ to $\{F_{IDS}, C_{IDS}\}$ **do**
4: Calculate B_i, $D_i^{vCPU}(B_i)$ and $D_i^{vMemory}(B_i)$.
5: **if** $(D_i^{vCPU}(B_i) \leq vCPU_i^{max})$ *and* $(D_i^{vMemory}(B_i) \leq vMemory_i^{max})$ **then**
6: **continue**
7: **else**
8: Forward $\frac{\Delta B_i}{Number\ of\ neighbors}$ to each Fog-IDS or Cloud-IDS neighbor, where $\Delta B_i = B_i - B_i^{max}$.
9: **end if**
10: Extract *features* from remaining data amount.
11: **if** $i \in F_{IDS}$ **then**
12: Feed inputs into a *moderate* IDS.
13: **else**
14: Feed inputs into a *compute-intensive* IDS.
15: **end if**
16: Get IDS's *outcome*.
17: **if** *outcome* = *normal* **then**
18: **continue**
19: **else if** *outcome* = *malicious* **then**
20: Call **Policy creation and enforcement**
21: **else if** *outcome* = *unknown* **then**
22: **if** $i \in F_{IDS}$ **then**
23: Forward to a connected Cloud-IDS node.
24: **else**
25: Send a report to the cloud administrator.
26: **end if**
27: **end if**
28: **end for**

Algorithm 3 Proposed Database Update Scheme

1: $N = \{E_{IDS}, F_{IDS}, C_{IDS}\}$ ← Set of all IDS nodes.
2: New_i ← New traffic patterns collected by an IDS node i that is not sent to any nodes before.
3: T_i ← Regular period of time (minutes) for updating connected nodes of an IDS node i.
4: $Time\_Counter_i$ ← Time counter (minutes) for database exchange of an IDS node i.
5: **for** $i = 1$ to N **do**
6: **if** $Time\_Counter_i \geq T_i$ **then**
7: **if** $B_i < B_i^{max}$ **then**
8: **if** $i \in E_{IDS}$ **then**
9: Send New_i to connected Fog-IDS node.
10: **else if** $i \in F_{IDS}$ **then**
11: Send New_i to connected Fog-IDS and Cloud-IDS nodes.
12: **else if** $i \in C_{IDS}$ **then**
13: Send New_i to connected Cloud-IDS nodes.
14: **end if**
15: $Time\_Counter_i \leftarrow 0$
16: **end if**
17: **else**
18: $Time\_Counter_i \leftarrow Time\_Counter_i + 1$
19: **end if**
20: **end for**

continuously collects IoT traffic flow statistics information passing the associated SDN-based IoT gateway device. For example, the gateway device is an OpenFlow switch, then the collection data of IoT traffic is the switch's flow-table data, including all raw information of flow-tables [45]. Moreover, the collected amount of data represented by a corresponding bandwidth value, B_i, are forwarded to the Edge-IDS node i for processing. Afterward, the Edge-IDS checks its current resource usage ($vCPU$, $vMemory$) and compares with the condition mentioned in Equation (4) (see line 6 in Algorithm 1). If the condition is satisfied, it means the Edge-IDS node i can handle all the incoming data. Otherwise, it decides to load balance a part of the incoming data to a connected Fog-IDS with an exact amount, see line 9 in Algorithm 1.

Next, the Edge-IDS node i conducts feature engineering and extraction to take out values of desired features; the Edge-IDS forms these values into inputs and feeds them into its classification algorithm [27]. There are two possibilities of outcomes from the machine learning-based detection engine on an input: obvious (*normal* or *malicious*) and *unknown* traffic patterns. For obvious decisions, the Edge-IDS can take action/policy immediately to stop malicious traffic flows passing the gateway while keeping normal traffic flows. Since there are several types of network-related attacks which can happen in IoT networks (see section II-B); therefore, we will adequately address the policy creation and enforcement in the *SeArch* system later. For *unknown* patterns, it means the Edge-IDS i cannot decide on this input because of any reasons such as lacking key features or unseen type of traffic, and then these patterns will be forwarded to a superior Fog-IDS node for security checking with more advanced knowledge. Algorithm 1 summarizes the operation of Edge-IDS nodes.

Regarding the Fog-IDS nodes, they perform detection tasks for unknown and load-balanced traffic patterns from the edge nodes and among the fog nodes. This is reasonable due to a low latency between the edge and fog computing levels (see section II-A.2 and [12], [19]), while only unhandled data is transferred to the Cloud-IDS nodes for the most advanced security analyses based on deep learning. Finally, polices are loaded to SDN controllers that can apply these policies to corresponding SDN-based IoT gateway devices to drop/block abnormal traffic flows. Similar processes are applied for the Cloud-IDS nodes in runtime operation. All mentioned processes can be summarized in Algorithm 2.

3) DATABASE UPDATE SCHEME

One of the essential factors encouraging the performance improvement of the *SeArch* solution is the database synchronization and update scheme among IDS nodes. The reason is that the network traffic behavior of IoT networks is always dynamic time by time; hence, it is needed to update databases of IDS nodes and train machine learning algorithms with updated data in order to produce more accurate decisions. Therefore, we propose a scheme for database update among nodes of the system, which is accurately described in Algorithm 3. From the proposed scheme, we can see that Edge-IDS nodes update Fog-IDS nodes, Fog-IDS nodes share new data with Fog-IDS and Cloud-IDS nodes, and Cloud-IDS nodes only update among them.

Generally, IoT traffic is considered as heterogeneous from a macro perspective; however, according to [46] it is group-specific from the perspective of each local IoT network. Thus, our proposed arrangement allows the detection engine at each SDN-based IoT gateway to more efficiently classify group-specific (local) IoT network traffic than mixed several traffic types. Even if there is an unseen traffic pattern, the Edge-IDS can send it to the Fog-IDS for further checking, and because the fog nodes have more knowledge of the traffic compared to the edge nodes by sharing and updating databases together. Consequently, the unseen traffic pattern will be verified very soon by the designated Fog-IDS node, then a policy can be issued to drop/block the traffic flow if the unseen data is recognized as an attack pattern. In conclusion, the proposed Algorithm 3 can allow the system to improve the performance of *source-based* detection for both group-specific and heterogeneous traffic at local IoT networks.

V. IMPLEMENTATION AND EXPERIMENTS

A. DEPLOYMENT SETUP

To verify the novelty of the *SeArch* architecture, we have implemented the proposed NIDS framework, as illustrated in Figure 5. In which, we leverage the use of a distributed SDN emulator, MaxiNet [33], and a docker container-based framework - Containernet [47], in order to emulate a distributed SDN-based networks through 03 physical machines. In each machine, we set up a small scale network with some OpenvSwitches running as SDN-based IoT gateways associated with Edge-IDS ($E1$-$E9$[5]) nodes, and 12 container-based hosts (d1-d12[6]) are distributed as shown in Figure 5. ONOS [48] is chosen as the SDN controller with an Fog-IDS ($F1$-$F3$[7]) in the application layer. GRE tunnels are set up between machines to transfer data among SDN-based networks. In addition, 02 physical machines are used for formulating the cloud environment with 02 Cloud-IDS ($C1$-$C2$[8]) nodes, and these physical machines are connected to others using two L2 switches. A part of Figure 5 shows the NIDS node graph connecting all nodes. Five machines are all Intel Core i7-4790 3.60 GHz and 16 GB DDR3 1600 MHz.

Note that for ease of implementation, we assume that we already applied two functions $ESRM\_func(N)$ and $FSP\_func(i, j)$ (see section IV-C) to solve the resource allocation and the deployment placement issues.

B. EXPERIMENTS

1) MACHINE LEARNING-BASED DETECTION AND TRAINING DATASETS

As discussed above in sections III-B, IV-B, and IV-D.1, we now have to select the machine learning algorithms for detection engines in the *SeArch* architecture. Again, for an ease of deployment, we utilize the *lightweight* traffic classifier [35] (Support Vector Machine - SVM) for Edge-IDS, the *moderate* traffic classifier [36] (Self Organizing Map - SOM) for Fog-IDS, and the *compute-intensive* traffic classifier [37] (Stacked Autoencoder Deep Learning approach - SAE) for Cloud-IDS, respectively. Nevertheless, other machine learning-based detection techniques are absolutely feasible to apply to the proposed system.

Moreover, we thoroughly investigated the network-related threats in IoT networks in section II-B, including eavesdropping, denial-of-service (DoS/DDoS), spoofing and man-in-the-middle (MITM) attacks. However, it can be seen that to achieve sensitive information for conducting eavesdropping, spoofing, and MITM attacks, intruders must prior launch a probing attack to gather useful network or target information. Consequently, there are two common network-related cyber-attacks, including DoS/DDoS and Probe/Reconnaissance attacks which can harm IoT networks, generally. Furthermore, Table 3 illustrates the statistics of three well-known datasets for network intrusion detection system, CAIDA [38], [39], KDD Cup 1999 [43], and UNSW-NB15 [44]. From the above analyses, we decide to consider DoS/DDoS and Probe/Reconnaissance attacks in our experiments. Moreover, we separately implemented 02 types of SVM detection engine for both above threats in each Edge-IDS node ($E1 - E9$), while SOM and SAE algorithms allow us to examine many traffic types using one detection engine, for example in [17], [40], the SOM is trained with and classifies both ICMP and TCP DDoS traffic using a 2-dimensional map.

Initially, 30,000 data samples are extracted from three datasets in Table 3 with a wide range of features. Based on studies in [17], [40], we train Edge-IDS and Fog-IDS detection engines using simple features consisting of *number of flows*, *number of packet per flow*, *number of byte per flow*, *flow duration*, *growth of client ports*, and *protocol*. For deep learning-based Cloud-IDS nodes, we apply a set of 9 features for TCP traffic and a set of 6 features for ICMP traffic as the same in [37]. It is noted that we train IDS nodes for anomaly detection according to their corresponding IoT traffic group patterns, which is mentioned in the next section.

[5] $E1$-$E9$ stand for 09 Edge-IDS nodes
[6] d1-d12 stand for 12 docker container-based hosts
[7] $F1$-$F3$ stand for 03 Fog-IDS nodes
[8] $C1$-$C2$ stand for 02 Cloud-IDS nodes

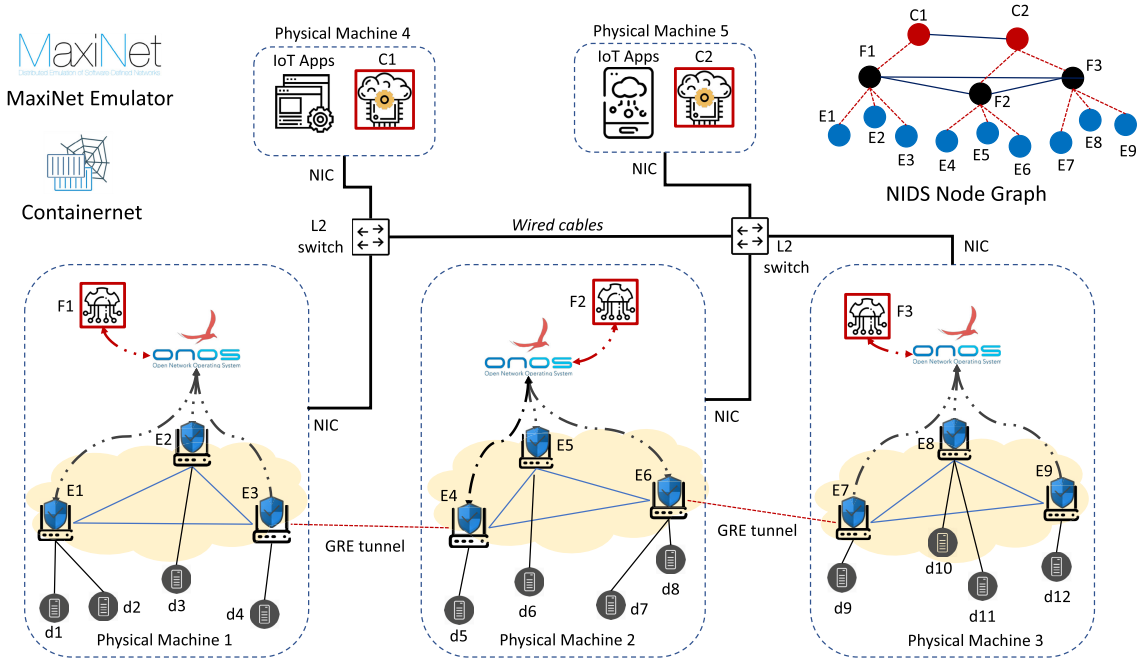

FIGURE 5. Deployment topology of the proposed *SeArch* architecture.

TABLE 3. Statistical distribution of CAIDA, KDD Cup 1999 and UNSW-NB15 data sets.

CAIDA data sets [38], [39]			
Traffic Types	TCP Protocol (%)	ICMP Protocol (%)	Other Protocols (%)
Normal (2015)	88.45	6.0	5.55
DDoS Attack (2007)	7.58	91.25	1.17
KDD Cup 1999 data set [43]			
Attack types	Training Patterns	Testing Patterns	Number of Features
Probe, DoS, U2R, R2L	45,927	7,458	41
UNSW-NB15 data set [44]			
Attack types	Normal Patterns	Attack Patterns	Number of Features
Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic Reconnaissance, Shellcode, Worms	2,218,761	321,283	49

2) IoT TRAFFIC GENERATION

Although there is a wide variety of IoT devices, we already discussed this in section IV-D.3; therefore, we can generalize the types of traffic into three common groups:

- *Sensor traffic*: IoT sensor devices generate this traffic group in a certain period with a low number of packets per traffic flow.
- *Monitor traffic*: This traffic type is generally referred to real-time applications, characterized by a small number of flows but a significant number of packets per flow.
- *Alarm traffic*: This traffic group is not easily described because alarm IoT devices only generate traffic when an abnormal event occurs. Accordingly, we assume this traffic group has a moderate amount regarding both the number of flows and the number of packets per generated flow.

As can be seen in Figure 5, we configure each of small scale SDN-based network to generate only one type of mentioned traffic groups as follows: *Sensor traffic* is for IoT devices d1 to d4, d5 to d8 are generating *Monitor traffic* and *Alarm traffic* is processed by d9 to d12, respectively. To do so, we utilize BoNeSi traffic generator tool [34], which provides various configurations of traffic generation, to generate HTTP-based traffic for normal IoT traffic. Regarding attack traffic generation, BoNeSi is also used to generate DoS/DDoS traffic, and Nmap scanning tool [49] is chosen in order to launch Probe/Reconnaissance attacks. These tools are installed in container-based IoT devices.

3) POLICY CREATION AND ENFORCEMENT

After the detection phase is done in any NIDS nodes, policy creation and enforcement play a key role in stopping attack traffic flows in the IoT networks. Due to two mentioned DoS/DDoS and Probe/Reconnaissance attacks; hence, we propose the following process to defend against or mitigate malicious traffic flows.

DoS/DDoS attacks: In [50], authors have carefully analysed the two most common DDoS attack types from the perspective of the SDN-based network. The first type (t_1) is to rely on the volume of packets or data coming from a source address which generates one or two flows with a high volume of packets in each flow; for example, Smurf and Fraggle, and ICMP flooding attacks [10]. The second type (t_2) relies on the volume of the number of flows from a source IP address in a short period (e.g., TCP SYN flooding [10]) and these flows may be kept alive during the attack (e.g. Low and Slow rate DDoS attack [51]). Accordingly, in the *SeArch* solution, if an attack is detected at an Edge-IDS node, this node should be allowed to install flow rules into the affected SDN-based IoT gateway for a fast reaction to attacks. Otherwise, a Fog-IDS or Cloud-IDS should inform the associated SDN controller to sends a *flow_mod* message [45] to the affected gateway device. For the first attack type, the policy is with a *drop* action and a preset *hard_timeout* value to drop every packet from the attacking source. Meanwhile, the sent *flow_mod* message is with a *delete* action to remove malicious flows in the second attack type. Besides, these IDS nodes can inform the forwarding engine of the SDN controller to drop *packet_in* messages of attacking sources which require for new flow installation at the gateway.

Probe/Reconnaissance attacks: Because of the less traffic volume generated from this kind of attacks, it could be challenging to mitigate these attacks. However, in this study, we suggest installing flow rules with a *source* IP matching field and *drop* action for the malicious sources in a preset *hard_timeout* value. By doing so, within a specific period, the source cannot access to the destination in any approaches due to the *drop* action that matches to every incoming packet from the abnormal source IP address.

4) ATTACK SCENARIOS

We now discuss about attack scenarios in our deployment.

DoS/DDoS attack scenarios: At first, we launch local DDoS attacks (L_{t_1} - ICMP Flooding and L_{t_2} - TCP SYN Flooding) in the IoT network including d5 to d8 containers, in which d5, d6, and d7 simultaneously flood d8 (an Apache webserver). After that, global DDoS attacks (G_{t_1} - ICMP Flooding and G_{t_2} - TCP SYN Flooding), malicious traffic is generated from all IoT devices, are conducted in order to break down the web service in d8 container.

Probe/Reconnaissance attack scenarios: We carry out a local scanning attack (L_{Probe}) from d5 to d8 by launching the Nmap tool. Similarly, a global scale probing attack (G_{Probe}) is launched from both d5 and d12 to d8 to achieve sensitive information about the destination device.

Note that we carry out 10 experiment trials with 500 seconds for each, and mentioned attacks are generated many times to the victim device during each trial.

5) RELATED SOLUTIONS FOR COMPARISON

To show the inspired performance of the *SeArch* architecture in anomaly detection and mitigation for SDN-based cloud IoT networks, we carry out experiments of the proposed solution in comparison with following other existing methods:

- *Distributed Edge-based Defense (DED)*: In references [16], [17], authors proposed to utilize the SDN-based IoT gateway device to detect and perform an appropriate response in case of an attack presence. Setting up this DED architecture in our testbed, machine learning-based detection engines or Edge-IDS nodes ($E1 - E9$) are placed at SDN-based IoT gateway devices located in three physical machines (1-3) as shown in Figure 5, and they are then trained individually without in any collaborations.
- *Centralized Fog-based Defense (CFD)*: References [12], [13] present similar approaches for anomaly detection based on the power of the fog computing and improved or combined intelligent classifiers. To achieve the CFD architecture setup, we deploy another physical machine with a centralized ONOS SDN controller which controls all SDN-based IoT gateway devices[9] in three physical machines (1-3) as shown in Figure 5. Afterward, we place a Fog-IDS node on top of this ONOS controller, in which the Fog-IDS collects traffic statistics information from all SDN-based IoT gateway devices and conducts its detection.
- *Centralized Fog and Cloud-based Defense (CFCD)*: Authors in reference [11] introduce a multi-level DDoS mitigation framework leveraging the use of both the fog and the cloud computing levels. Similarly, we utilize the above centralized CFD setup and then connect the Fog-IDS node to a Cloud-IDS node at another machine, e.g., the physical machine 4 in Figure 5. Next, the Fog-IDS collects traffic statistics information from all SDN-based IoT gateway devices and conducts its detection. Moreover, in case of overload, the Fog-IDS does a load balancing with the Cloud-IDS node for more advanced analyses.

Note that we implement these solutions [11]–[13], [16], [17] in our testbed according to the above description to keep their original novelties and functionalities.

TABLE 4. Anomaly detection performance indices.

Detection Rate	Precision	Accuracy	FAR
$\frac{TP}{TP+FN}$	$\frac{TP}{TP+FP}$	$\frac{TP+TN}{TP+FP+TN+FN}$	$\frac{FP}{TN+FP}$

VI. RESULT ANALYSIS

A. ANOMALY DETECTION PERFORMANCE

In our experiments, we measure four parameters: True Positive (TP) is the probability of abnormal traffic flows, which are classified as illegal flows. True Negative (TN) shows the probability of legitimate traffic flows that are trusted as normal flows. False Positive (FP) reflects the probability of normal traffic flows that are judged as abnormal flows,

[9]Note that, in this setup, there are no associated Edge-IDS nodes in SDN-based IoT gateways.

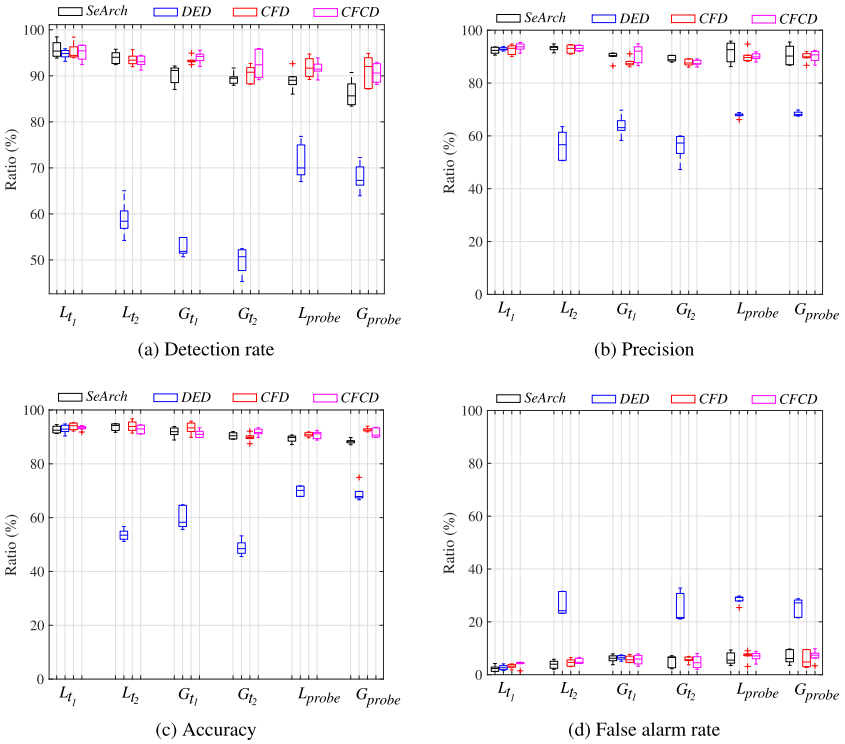

FIGURE 6. Anomaly detection performance comparison among 4 different solutions. (a) Detection rate. (b) Precision. (c) Accuracy. (d) False alarm rate.

and False Negative (FN) reveals the probability of attack traffic flows that are recognized as legal flows. According to the recorded results, we next calculate four important criteria including Detection rate, Precision, Accuracy, and False alarm rate (FAR) to evaluate the performance of our proposed model, as illustrated in Table 4. Detection rate is the ratio of correctly identified attack over the total amount of attacks happened in the networks. On the other hand, Precision represents the ratio of correctly identified attacks over the total amount of identification of attacks. False alarm rate shows the ratio of incorrectly identified attack over the total amount of incorrect identification, whereas Accuracy means how correct the detection engine is. All indices are measured in trials during different attacks.

As shown in Figure 6, it is clear that the *DED* solution achieves poor performances in terms of Detection rate, Accuracy, Precision, and False alarm rate for six different attack scenarios. This is understandable because Edge-IDS nodes could not effectively detect unknown malicious traffic patterns without any collaborations, and achieve irrelevant results when trained detection engine was updated during attack time. Contrary *SeArch*, *CFD*, and *CFCD* schemes produce similar results of all four evaluation criteria. The proposed *SeArch* is slightly lower regarding Detection rate since the system needs to exchange information to become converged and then achieve the most accurate decision for every traffic pattern. Meanwhile, it shows not much different in terms of Accuracy, Precision, and False alarm rate compared to *CFD* and *CFCD* approaches. To sum up, the *SeArch* architecture guarantees a remarkable level of anomaly detection performance in comparison with the centralized solutions, while completely outperforms the distributed approach with non-collaborations.

B. ATTACK MITIGATION PERFORMANCE

Attack mitigation plays a crucial role in securing every networked system; hence, we now analyze the results recorded from our experiments among four comparing solutions.

1) AVERAGE DETECTION TIME OF A NEW ATTACK PRESENCE IN THE TARGETED NETWORK

Firstly, we consider how fast a new attack is detected in the victim network. By generating unknown attacks of detection engines (e.g., TCP SYN Flooding and Probe/Reconnaissance) in the IoT network residing in the physical machine 2 (see Figure 5), we measure and calculate the average detection time of a new attack as depicted in Figure 7. It is evident that at first the *SeArch* mechanism would take a bit longer time compared to *CFD* and *CFCD* solutions to detect a new incoming attack in the victim network, this is because of collaboration time among IDS nodes and this leads to a less delay of new attack detection performed by the *SeArch* solution later on. Meanwhile, *DED* could not detect a new attack if Edge-IDS nodes do not update its detection brain to adapt to the current network traffic; therefore, we denote N/A (not available) in this case. In summary, as the *SeArch* architecture becomes converged, it can recognize a new attack pattern very quickly in comparison with centralized solutions.

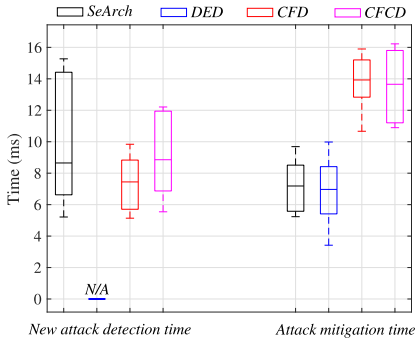

FIGURE 7. Average new attack detection time and average attack mitigation time in different solutions.

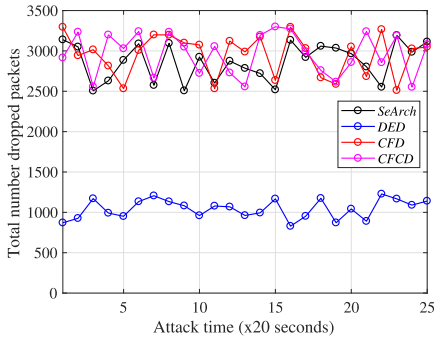

FIGURE 8. Number of dropped packets over attack time for different approaches.

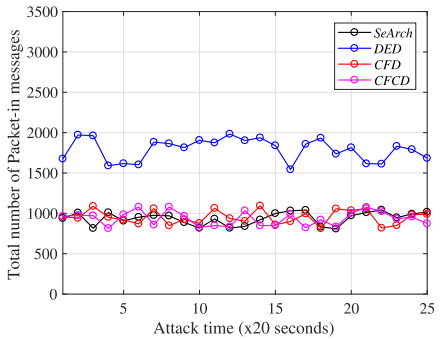

FIGURE 9. Number of packet_in messages to the SDN control plane during attack for different approaches.

2) AVERAGE ATTACK MITIGATION TIME

Furthermore, we look at how fast policy is conducted and implemented in the data plane since a detection alert is raised. It is clear that the *DED* solution only relies on source-based detection and mitigation; thus, it takes a very little time to implement policies if an attack is detected, as shown in Figure 7. Similarly, the *SeArch* solution also bases on *source-based* detection and mitigation as discussed earlier; thus, it is just a little slower than the *DED* scheme because of some unknown patterns at the beginning, and it absolutely outperforms the *CFD* and *CFCD* approaches in more quickly stopping attack traffic.

3) NUMBER OF DROPPED MALICIOUS PACKETS

To efficiently stop abnormal traffic in the IoT network, one key evaluation criterion is how many malicious packets are dropped during the attack. Therefore, we record the total number of dropped malicious packets when IoT networks are under several attacks, and results are illustrated in Figure 8. Because of the high degree of detection rate, *SeArch*, *CFD* and *CFCD* solutions conduct and implement policies as soon as an attack pattern is detected leading to a high number of dropped attack packets. In the case of *DED* scheme, it could not efficiently recognize abnormal traffic compared to other solutions, and it only dropped a low number of malicious packets during attacks as a result.

4) NUMBER OF PACKET_IN MESSAGES TO THE SDN CONTROL PLANE

Attack traffic not only can harm the data plane and IoT devices but also produce harmful effects on the control plane. In our experiments, we take the number of packet_in messages to the SDN controller into account to represent this judgement criterion. As depicted in Figure 9, due to a low detection rate of abnormal attacks in case of *DED* scheme; hence, it causes many flow mismatching events in SDN-based IoT gateways leading to a considerable volume of packet_in messages are generated and sent to ONOS controller. However, three remaining mechanisms including *SeArch*, *CFD*, and *CFCD* show an acceptable packet_in rate to the control plane, and this is due to a right level of malicious traffic detection and a fast implementation of policies in the data plane, which protects the SDN control plane from a packet_in flooding attack [52] in case of high rate attacks.

5) NUMBER OF CONDUCTED POLICIES AND TRAFFIC FLOW QUANTITY IN THE SDN-BASED IoT GATEWAYS

Finally, we pay attention to the number of conducted policies (rules) in the data plane devices to judge our proposed architecture in attack mitigation performance. As can be seen in Figure 10, the *SeArch*, *CFD*, and *CFCD* methods create and enforce policies more than twice on average in comparison with the *DFD* solution. As we analyzed above, a high detection rate implies a high number of generated policies, and then they are implemented to the SDN-based IoT gateway devices; therefore, the recorded results are reasonable. In addition, Figure 11 represents the total number of flow entries in SDN-based IoT gateway devices during the attack time. We can see that a higher number of conducted polices results in many deleted malicious traffic flows; otherwise, the SDN-based gateways must maintain a huge number of flow entries in their flow-tables. In conclusion, the *SeArch* architecture again achieves remarkable results in protecting the data plane from being flooded by saturation attacks.

C. SYSTEM OVERHEAD

To evaluate a new security architecture, the system overhead criteria should also be considered; hence, we provide analyses about how the *SeArch* architecture can deal with bottleneck problems and overhead of collaboration at runtime operation.

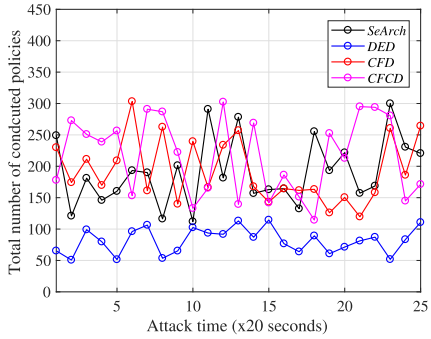

FIGURE 10. Number of conducted policies in the SDN-based IoT gateways during attack for different approaches.

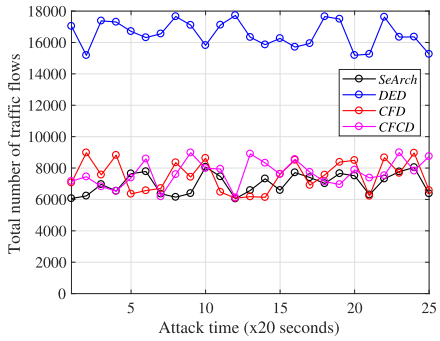

FIGURE 11. Total number of traffic flows in the whole emulated network under a massive DDoS attack - G_{t_2}.

1) PERFORMANCE BOTTLENECK HANDLING

As illustrated in Figure 12, we measure the average CPU utilization[10] of three IDS node levels when a massive DDoS attack - G_{t_2}, which is the most dangerous DDoS attack types, is launched. The IDS nodes of the *SeArch* proposal always consume the least computational resources in Edge and Fog computing levels. Only *CFCD-Cloud* nodes have a similar resource consumption degree with *SeArch-Cloud* nodes. Therefore, the *SeArch* solution saves the most computational resources on average, when the IoT networks are under a massive DDoS attack. Accordingly, the performance bottleneck issues are minimized in the case of the *SeArch* architecture in comparison with existing solutions.

2) OVERHEAD OF THE SYSTEM COLLABORATION

Finally, we examine the overhead of the system collaboration when the network is under different cyber attacks. We record the amount of data generated by load balancing and database update events in comparison with the total generated data in the *SeArch* architecture. As depicted in Figure 13, G_{t_2} and G_{probe} attacks produce just around 15.0 % of the total handling data for the system collaboration tasks. Meanwhile, other attacks only account for less than 8.5 % of the total data amount. In particular, L_{t_2} and L_{probe} create a moderate level of traffic flow quantity in a local network scale resulting

[10]We observe the CPU usage of the core running the security application. And, all machines are Intel Core i7-4790 (4 cores) 3.60 GHz and 16 GB DDR3 1600 MHz.

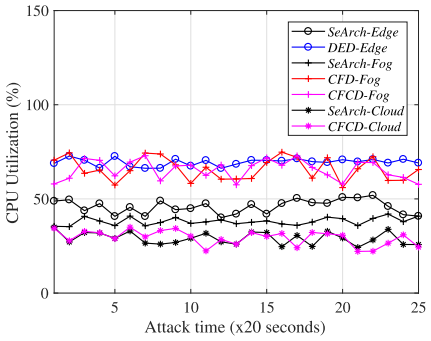

FIGURE 12. Average CPU utilization of IDS nodes of different solutions when a massive DDoS attack - G_{t_2} is launched.

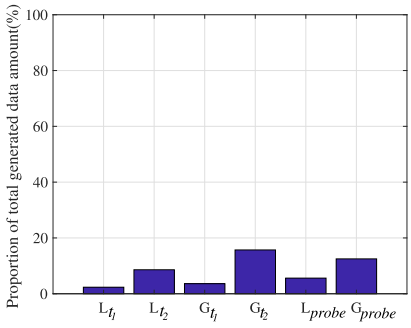

FIGURE 13. Data amount of load balancing and database update events compared to the total generated data of the *SeArch* system during different attacks.

in a low proportion requiring for the system collaboration overhead. L_{t_1} and G_{t_1} attacks generate little number flow entries in the SDN-based gateways; hence, the processing data amount is quite small, and the overhead mostly comes from the database update events. In summary, the overhead of the system collaboration is unnoticeable compared to the total data amount produced by the system operation.

D. DISCUSSION

Through comprehensive analyses above, one can see that the *SeArch* solution completely outperforms the *DED* scheme in terms of anomaly detection performance and most criteria of attack mitigation by using an efficient collaboration scheme. However, for attack mitigation time, the *SeArch* needs a little longer time than the *DED* method to mitigate attack traffic at the beginning due to the collaboration process, and it is expected to achieve better performance for a long run.

The *SeArch* framework completely surpasses the *CFD* and *CFCD* methods regarding attack mitigation time and dealing with performance bottleneck problems by leveraging a good *source-based* location to place detection engines. Nonetheless, with regard to the detection performance, i.e., detection rate, precision and accuracy, and mitigation attack criteria including number of dropped malicious packets, number of packet_in messages, number of conducted polices, and number of traffic flow rules, the *SeArch* have similar performance results with the *CFD* and *CFCD* methods.

From our above discussion, it proves that the proposed *SeArch* architecture effectively benefits advantages from

other existing solutions while altogether avoiding their drawbacks in order to defense network-related cyber-attacks in SDN-based cloud IoT networks. Besides, based on our proposed functions in section IV-C, the *SeArch* architecture gives the cloud administrator an efficient mechanism for resource optimization in practical deployment.

VII. CONCLUSION

In this paper, we propose a new security architecture, *SeArch*, representing a collaborative and intelligent NIDS system in SDN-based cloud IoT networks, in which an arrangement of three layers of IDS nodes (Edge-IDS, Fog-IDS, and Cloud-IDS) is introduced with an effective collaboration among nodes. This architecture leverages the use of machine learning/deep learning for intelligently detecting network-related threats from IoT devices. A novel system resource optimization and an optimal path selection scheme are proposed to bring benefits to the resource management and the overhead of communication of the proposed solution. In comparison with existing solutions, the *SeArch* solution achieves a remarkable anomaly detection performance, i.e., around 95.5% on average of detection rate, accuracy, and precision, which is same to results obtained by the *CFD* and *CFCD* methods, while providing a right level of attack mitigation, i.e., only 7.0 *ms* on average in attack mitigation time, and tackling performance bottleneck problems as same as the *DED* scheme does. Additionally, the *SeArch* architecture presents only a minor overhead of the system collaboration, i.e., from 8.5% to 15.0%. As our future development, we plan to investigate other machine learning/deep learning algorithms and cyber attacks with a more massive amount of data sets and various/heterogeneous traffic types in the proposed architecture.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.

[2] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on Internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.

[3] H. Arasteh, V. Hosseinnezhad, V. Loia, A. Tommasetti, O. Troisi, M. Shafie-khah, and P. Siano, "IoT-based smart cities: A survey," in *Proc. IEEE 16th Int. Conf. Environ. Elect. Eng. (EEEIC)*, Jun. 2016, pp. 1–6.

[4] M. Alaa, A. A. Zaidan, B. B. Zaidan, M. Talal, and M. L. M. Kiah, "A review of smart home applications based on Internet of Things," *J. Netw. Comput. Appl.*, vol. 97, pp. 48–65, Nov. 2017.

[5] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[6] Cisco Systems. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2017–2022*. Accessed: Mar. 15, 2019. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html

[7] H. Aldowah, S. U. Rehman, and I. Umar, "Security in Internet of Things: Issues, challenges and solutions," in *Recent Trends in Data Science and Soft Computing*, F. Saeed, N. Gazem, F. Mohammed, and A. Busalim, Eds. Cham, Switzerland: Springer, 2019, pp. 396–405.

[8] 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.

[9] V. Varadharajan and U. Tupakula, "Security as a service model for cloud environment," *IEEE Trans. Netw. Service Manage.*, vol. 11, no. 1, pp. 60–75, Mar. 2014.

[10] I. Farris, T. Taleb, Y. Khettab, and J. S. Song, "A survey on emerging SDN and NFV security mechanisms for IoT systems," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 812–837, 1st Quart., 2018.

[11] Q. Yan, W. Huang, X. Luo, Q. Gong, and F. R. Yu, "A multi-level DDoS mitigation framework for the industrial Internet of Things," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 30–36, Feb. 2018.

[12] Q. Shafi, A. Basit, S. Qaisar, A. Koay, and I. Welch, "Fog-assisted SDN controlled framework for enduring anomaly detection in an IoT network," *IEEE Access*, vol. 6, pp. 73713–73723, Nov. 2018.

[13] J. Li, Z. Zhao, R. Li, H. Zhang, and T. Zhang, "Ai-based two-stage intrusion detection for software defined iot networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2093–2102, Nov. 2018.

[14] A. S. da Silva, J. A. Wickboldt, L. Z. Granville, and A. Schaeffer-Filho, "ATLANTIC: A framework for anomaly traffic detection, classification, and mitigation in SDN," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2016, pp. 27–35.

[15] M. T. Kakiz, E. Ozturk, and T. Cavdar, "A novel sdn-based iot architecture for big data," in *Proc. Int. Artif. Intell. Data Process. Symp. (IDAP)*, Sep. 2017, pp. 1–5.

[16] 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.

[17] T. V. Phan, N. K. Bao, and M. Park, "Distributed-SOM: A novel performance bottleneck handler for large-sized software-defined networks under flooding attacks," *J. Netw. Comput. Appl.*, vol. 91, pp. 14–25, Aug. 2017.

[18] Y. Li, X. Su, J. Riekki, T. Kanter, and R. Rahmani, "A SDN-based architecture for horizontal Internet of Things services," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–7.

[19] P. J. Escamilla-Ambrosio, A. Rodríguez-Mota, E. Aguirre-Anaya, R. Acosta-Bermejo, and M. Salinas-Rosales, "Distributing computing in the Internet of Things: Cloud, fog and edge computing overview," in *NEO 2016*. Cham, Switzerland: Springer, 2018, pp. 87–115.

[20] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," ETSI, Sophia Antipolis, France, ETSI White Paper 11, 2015, no. 11, pp. 1–16.

[21] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[22] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[23] P. M. Mell and T. Grance, "The NIST definition of cloud computing," NIST, Gaithersburg, MD, USA, Tech. Rep. Sp 800-145, 2011.

[24] Q. Xu, P. Ren, H. Song, and Q. Du, "Security enhancement for IoT communications exposed to eavesdroppers with uncertain locations," *IEEE Access*, vol. 4, pp. 2840–2853, 2016.

[25] A. Mukaddam, I. Elhajj, A. Kayssi, and A. Chehab, "IP spoofing detection using modified hop count," in *Proc. IEEE 28th Int. Conf. Adv. Inf. Netw. Appl.*, May 2014, pp. 512–516.

[26] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of Things security: A survey," *J. Netw. Comput. Appl.*, vol. 88, pp. 10–28, Jun. 2017.

[27] F. Restuccia, S. D'Oro, and T. Melodia, "Securing the Internet of Things in the age of machine learning and software-defined networking," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4829–4842, Jun. 2018.

[28] M. Ojo, D. Adami, and S. Giordano, "A SDN-IoT architecture with NFV implementation," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2016, pp. 1–6.

[29] M. Nobakht, V. Sivaraman, and R. Boreli, "A host-based intrusion detection and mitigation framework for smart home IoT using OpenFlow," in *Proc. 11th Int. Conf. Availability, Rel. Secur. (ARES)*, Sep. 2016, pp. 147–156.

[30] M. A. Alsheikh, S. Lin, D. Niyato, and H. P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1996–2018, 4th Quart., 2014.

[31] T. T. T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 56–76, 4th Quart., 2008.

[32] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[33] P. Wette, M. Dräxler, A. Schwabe, F. Wallaschek, M. Zahraee, and H. Karl, "Maxinet: Distributed emulation of software-defined networks," in *Proc. Netw. Conf. (IFIP)*, Jun. 2014, pp. 1–9.
[34] M. Go. (2019). *The DDoS Botnet Simulator—BoNeSi*. Accessed: Mar. 15, 2019. [Online]. Available: http://sourceforge.net/projects/loic/
[35] T. V. Phan, T. Van Toan, D. Van Tuyen, T. T. Huong, and N. H. Thanh, "OpenFlowSIA: An optimized protection scheme for software-defined networks from flooding attacks," in *Proc. IEEE 6th Int. Conf. Commun. Electron. (ICCE)*, Jul. 2016, pp. 13–18.
[36] R. Braga, E. Mote, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Proc. IEEE Local Comput. Netw. (LCN)*, Denver, CO, USA, Oct. 2010, pp. 408–415.
[37] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based DDoS detection system in software-defined networking (SDN)," *EAI Endorsed Trans. Secur. Saf.*, vol. 4, no. 12, pp. 1–12, 2017.
[38] CAIDA. *The CAIDA UCSD Anonymized Internet Traces 2015*. Accessed: Mar. 15, 2019. [Online]. Available: http://www.caida.org
[39] CAIDA. *The CAIDA DDoS Attack 2007*. Accessed: Mar. 15, 2019. [Online]. Available: http://www.caida.org/data/passive/ddos-20070804-dataset.xml/
[40] T. V. Phan, N. K. Bao, and M. Park, "A novel hybrid flow-based handler with DDoS attacks in software-defined networking," in *Proc. Int. IEEE Conf. Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People, Smart World Congr. (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, Jul. 2016, pp. 350–357.
[41] L. Cherkasova and R. Gardner, "Measuring CPU overhead for I/O processing in the Xen virtual machine monitor," in *Proc. Annu. Conf. USENIX Annu. Tech. Conf.*, 2005, p. 24.
[42] D. G. Luenberger and Y. Ye, "Transportation and network flow problems," in *Linear and Nonlinear Programming*. New York, NY, USA: Springer, 2008, pp. 145–179.
[43] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. 2nd IEEE Int. Conf. Comput. Intell. Secur. Defense Appl. (CISDA)*, Jul. 2009, pp. 53–58.
[44] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. IEEE Military Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[45] *OpenFlow Switch Specification Version 1.5.1*. Accessed: Mar. 15, 2019. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf
[46] F. Alam, R. Mehmood, I. Katib, N. N. Albogami, and A. Albeshri, "Data fusion and IoT for smart ubiquitous environments: A survey," *IEEE Access*, vol. 5, pp. 9533–9554, 2017.
[47] M. Peuster, H. Karl, and S. van Rossem, "MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2016, pp. 148–153.
[48] (2019). *Description ONOS Controller*. Accessed: Mar. 15, 2019. [Online]. Available: https://onosproject.org/
[49] (2019). *Nmap Security Scanner*. Accessed: Mar. 15, 2019. [Online]. Available: https://nmap.org/
[50] T. V. Phan and M. Park, "Efficient distributed denial-of-service attack defense in SDN-based cloud," *IEEE Access*, vol. 7, pp. 18701–18714, Jan. 2019.
[51] T. A. Pascoal, Y. G. Dantas, I. E. Fonseca, and V. Nigam, "Slow TCAM exhaustion DDoS attack," in *Proc. SEC*, 2017, pp. 17–31.
[52] H. Wang, L. Xu, and G. Gu, "FloodGuard: A DoS attack prevention extension in software-defined networks," in *Proc. 45th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2015, pp. 239–250.

TRI GIA NGUYEN (M'17) received the B.Ed. degree from the Hue University of Education, Vietnam, in 2011, the M.Sc. degree from Duy Tan University, Danang, Vietnam, in 2013, and the Ph.D. degree from Khon Kaen University, Thailand, in 2017, all in computer science. He is currently a Lecturer with the Faculty of Information Technology, Duy Tan University. His research interests include the Internet of Things, sensor networks, wireless communications, wireless energy harvesting networks, mobile computing, computer systems, software defined networks, network functions virtualizations, and network security.

TRUNG V. PHAN (M'19) received the B.S degree in electronics and telecommunications from the Hanoi University of Science and Technology (HUST), Vietnam, in 2015, and the M.S degree in information communication technology from Soongsil University, Seoul, South Korea, in 2017. He is currently a Research Staff with the Faculty of Electrical Engineering and Information Technology, Technische Universität Chemnitz, Germany. His research interests include deep reinforcement learning, federated learning, software defined networks, network functions virtualization, the Internet of Things, and network security.

BINH T. NGUYEN is currently pursuing the undergraduate degree in information technology with the Sector of International Education, Hanoi University of Science and Technology (HUST). Since 2017, he has been a Student Research Assistant with the Future Internet Laboratory, HUST. His research interests include the Internet of Things, software defined networks, and network security.

CHAKCHAI SO-IN (SM'14) received the Ph.D. degree in computer engineering from Washington University at St. Louis, MO, USA, in 2010. He has interned at CNAP-NTU, Singapore, Cisco Systems, WiMAX Forums, and Bell Labs, USA. He is currently a Professor with the Department of Computer Science, Khon Kaen University. He has authored over 100 publications and ten books, including some in the IEEE JSAC, the IEEE Magazines, and Computer Network/Network Security Labs. His research interests include mobile computing, wireless sensor networks, signal processing, and computer networking and security. He has served as an Editor for SpringerPlus, PeerJ, and ECTI-CIT and as a Committee Member for many conferences and journals, such as GLOBECOM, ICC, VTC, WCNC, ICNP, ICNC, PIMRC, the IEEE Transactions, the IEEE Letters/Magazines, and the *Journal of Computer Networks and Communications*.

ZUBAIR AHMED BAIG is currently a Senior Lecturer in cybersecurity with the School of Information Technology, Deakin University, Melbourne, VIC, Australia. He has authored over 50 journal and conference articles and book chapters. His research interests include cybersecurity, artificial intelligence, smart cities, and the Internet of Things. He has served on numerous technical program committees for international conferences and has delivered numerous keynote talks on cybersecurity. He is currently serving as an Editor for the *IET Wireless Sensor Systems Journal* and *PSU Research Review Journal*.

SURASAK SANGUANPONG received the B.Eng. and M.Eng. degrees in electrical engineering from Kasetsart University, in 1985 and 1987, respectively. He is currently an Associate Professor with the Department of Computer Engineering and the Director of the Applied Network Research Laboratory, Kasetsart University. His research interests include network measurement, Internet security, and high-speed networking.