IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

IEEE SYSTEMS, MAN AND CYBERNETICS SOCIETY SECTION

Received 23 March 2023, accepted 10 April 2023, date of publication 13 April 2023, date of current version 19 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3266826

RESEARCH ARTICLE

MCAD: A Machine Learning Based Cyberattacks Detector in Software-Defined Networking (SDN) for Healthcare Systems

LAILA M. HALMAN, (Student Member, IEEE), AND MOHAMMED J. F. ALENAZI, (Senior Member, IEEE)

Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11495, Saudi Arabia

Corresponding author: Mohammed J. F. Alenazi (mjalenazi@ksu.edu.sa)

The authors extend their appreciation to Researcher Supporting Project number (RSPD2023R582), King Saud University, Riyadh, Saudi Arabia.

ABSTRACT The healthcare sector deals with sensitive and significant data that must be protected against illegitimate users. Software-defined networks (SDNs) are widely used in healthcare systems to ensure efficient resource utilization, security, optimal network control, and management. Despite such advantages, SDNs suffer from a major issue posed by a wide range of cyberattacks, due to the sensitivity of patients' data. These attacks diminish the overall network performance, and can cause a network failure that might threaten human lives. Therefore, the main goal of our work is to propose a machine learning-based cyberattack detector (MCAD) for healthcare systems, by adapting a layer three (L3) learning switch application to collect normal and abnormal traffic, and then deploy MCAD on the Ryu controller. Our findings are beneficial for enhancing the security of healthcare applications by mitigating the impact of cyberattacks. This work covers the testing of MCAD using a wide spectrum of both ML algorithms and attacks, and provides a performance comparison for every pair of ML algorithms/attacks to illustrate the strengths and weaknesses of different algorithms against a specific attack. The MCAD shows impressive performance, achieving an F1-score of 0.9998 and of 0.9882 on normal and attack classes, respectively, which implies a high level of reliability. MCAD also achieved 5,709,692 samples per second on throughput, which reflects a high-performance real-time system with respect to complexity. Additionally, it showed a positive impact on the network KPIs by increasing the throughput by 609%, and decreasing delay and jitter by 77% and 23%, respectively, compared to attack results.

INDEX TERMS Network resilience, network management, intrusion detection system (IDS), software defined networking, healthcare, machine learning.

I. INTRODUCTION

In the last few years, SDNs have been extensively used in different fields, principally thanks to their advantages as reliable network technology that allows controlling and managing a network by disaggregating both control and data planes. In contrast to traditional networks, where the network simply has application awareness, the SDN architecture provides additional information about the condition of the entire network from the controller to its applications [1].

Following the recent high-paced progress in information and communications technologies (ICT), healthcare establishments have begun to employ numerous infrastructure factors of the same types of off-the-shelf technologies, applications, and procedures employed by companies from other sectors. This situation was expected, due to the ability of networked or Internet-connected medical tools to increase the effectiveness of asset management, communications, and electronic health records, among other requirements, which reduces cost (e.g., cost related to treatments and monitoring).

A large amount of expenditure is expected from healthcare establishments for network technologies in the coming years, though expenses for hospital equipment are expected

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

to decrease by 15 to 30% [2]. Furthermore, the safety of systems and devices, together with user data confidentiality are the two factors that are primarily taken into account in the majority of information systems, since confidentiality and safety are crucial in a healthcare context due to the exacting requirements of the industry. Therefore, it is important that the current McAfee record highlighted that networked medical tools may reveal security gaps in the attempt by the medical industry to incorporate all the technical elements related to networked infrastructure and operational controls [3].

Besides the susceptibility of information in healthcare networks, the intricacy, quantity, and variety of tools, particularly networked medical devices (e.g., wireless pacemakers) creating this infrastructure, networks will be exposed to a wider variety of privacy risks and security [4], [5]. During the COVID-19 pandemic, the number of attacks has increased five times. Consequently, 90% of healthcare providers have been subjected to data violations [6]. As proven in recent ransomware incidents [7], the healthcare industry is particularly vulnerable to cyberattacks, which may be attributable to confidentiality breaches (e.g., leaked or comprised sensitive medical records), incidental errors, or deliberate and extensive interference (e.g., caused by flawed construction, use, or function). Researchers have recently begun to explore the prospect of using SDN in healthcare establishments due to the ability of SDN to abstract network policy from network devices [8].

In relation to cyber security in healthcare establishments, SDNs could be employed as protection for medical networks against various harms (e.g., denial-of-service (DoS) and probe attacks). However, in common with current or conventional security resolutions such as intrusion identification and precaution systems or centralized protection methods, SDN solutions do not offer protection to the data and system from insider threats [9]. To illustrate, 92% of healthcare establishments revealed issues faced by their companies due to insider threats and needed appropriate resolutions for protection [10]. This condition makes it important to design functional solutions to reduce insider threats.

The main contributions of our work are:

1) To adapt an L3 learning switch application to collect normal and abnormal traffic.
2) To generate a dataset that contains both normal and abnormal traffic.
3) To adapt an efficient detection model based on machine learning.
4) To deploy MCAD on a Ryu controller.
5) To prove the ability of the MCAD model to deal with attacks and save network resources by measuring key performance indicators (KPIs), such as throughput, jitter, and delay of the network, using a Distributed Internet traffic generator tool (D-ITG).

Therefore, in this article, we propose the MCAD, an efficient low-complexity approach to detect cyberattacks for healthcare systems. This approach uses machine learning in SDNs as an efficient technique for detecting threats against healthcare system networks. The approach is analyzed based on network KPIs. The effectiveness of the proposed system is demonstrated and tested.

The sections of this paper are arranged as follows. In Section II, we introduce a description of the background and previous work. In section III, we describe the main aspects of our proposed approach, including the machine learning algorithms and tools used for implementing it, datasets used for training and testing, as well as the network topologies used, emulation environment, and performance metrics. Section IV presents the emulation results and a comprehensive evaluation after showing the experimental evaluation methodology. Finally, Section V offers conclusions and suggests future work.

II. BACKGROUND AND RELATED WORKS

A lot of effort has been put into the field of traffic classification and IDS using different techniques, including machine learning and artificial intelligence. In this section, we present an introduction to the SDN architecture, followed by a discussion of the significance of SDN in the healthcare sector. After that, we illustrate the SDN security challenges. Finally, we discuss recent research works on IDSs for SDNs.

A. BACKGROUND

1) SOFTWARE-DEFINED-NETWORKING

A common SDN consists of numerous control entities and programmable switches to move networking functionalities into a user-defined interface. In general, the SDN architecture offers numerous benefits, including management, dynamics, and cost-effectiveness. In contrast to traditional networks, SDN is able to employ more elements to its architecture, such as the addition of any software that could function within a server or a central processing unit (CPU). This action enables the transfer of network performance to a systematic software interface that can be deployed on the control plane [11]. Based on Figure 1, which presents the SDN three-layer architecture, the first layer denotes the application layer, whose role is to enforce strategies via the northbound application programming interfaces (APIs) sustained by the control layer (i.e., second layer). In contrast, southbound APIs are employed to support exchanges between the third layer (i.e., infrastructure layer) and the control layer. SDN controllers are able to function as a strategic control point in the network in order to control processes for implementations, policy engines, and switches.

The centralized architecture of network management does not require the user's acknowledgement of the underlying physical network and network topology, which could lead to a significant reduction of workload to manage an entire network that includes a wide range of operations. The use of SDN controller allows users or companies to gain independent control of the entire network from logical and single points. To illustrate, centralized control in SDNs allows

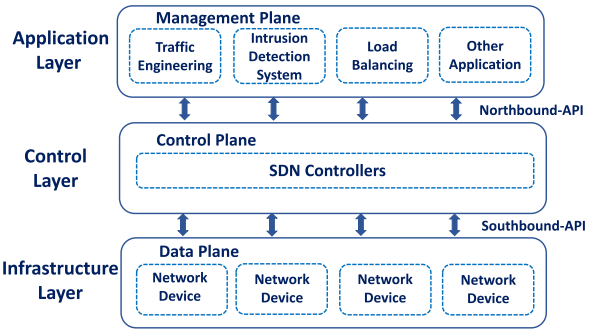

FIGURE 1. SDN architecture.

users to immediately supervise and control network processes at the application layer and swiftly implement new services or applications. Several recent controllers are available, including Ryu, POX, OpenKilda, Trema, OpenDaylight, and Floodlight. We developed MCAD on the Ryu controller because it has a number of advantages, including TLS support, open-source code, virtualization, GUI, restful API, modularity, productivity, documentation, platform support, age, OpenFlow support, and OpenStack Neutron compatibility [12].

Overall, users are able to employ general network services (e.g., multicast and routing) to acquire either organizational or individual objectives. Users are able to employ related APIs between applications and controller, followed by abstracting the network through leveraging network services without having knowledge of the application's particulars [13], [14]. The presence of new flow at a switch in the SDN allows the switch to determine the forwarding path by transmitting a routing request to the centralized controller. Notably, the controller is responsible for generating a routing path and exchanging the forwarding rule through a secure channel with relevant switches. Following the rule acquisition, all correlated SDN switches can present an improvement to flow tables. Overall, an SDN can offer efficient management of the entire network and provide diverse advantages, such as secure cloud services and on-demand resources, due to its centralized control and global view. Meanwhile, devices are only required to follow guidelines from related SDN controllers, and the understanding of thousands of protocol requirements is not required. Moreover, the SDN is able to enhance resilience compared to conventional networks. For example, a convenient modification or reconfiguration can be conducted on the SDN controller to improve exchanges among various elements in comparison with hardware-based devices.

2) SDN IN HEALTHCARE SECTOR

A crucial factor in healthcare systems is the ability to consolidate the systems while retaining seamless communication between them. In general, the consolidation of traditional networks is challenging, considering that every network device may include hundreds of arrangements that need to be modified [14]. Therefore, an SDN is a practicable solution allowing the abstraction of network policy from network devices, eliminating device-level arrangements, and providing an open networking model to be consolidated. To illustrate, an SDN may be employed to present patients with data security and speed in the transfer of information from endpoint to endpoint [14], [15]. As defined in [16], "the SDN controller identifies the connectivity of the patient monitoring endpoint to the network. Forwarding entries are incorporated into the network switches, allowing connection of the endpoint solely to patient monitoring controller". Connection of the monitoring endpoint could be established at any point in the SDN switch network, as the SDN controller would perform automatic identification of the endpoint and connection of the ingress interface to the correlated virtual network, and offers mobility, safety, and dependability. Figure 2 shows a communication topology that uses SDN in the healthcare sector.

This architecture consists of an SDN controller, a series of OpenFlow switches, and a quantity of medical and client devices (e.g., personal computers and mobile devices). The OpenFlow specification offers a regulated method of applying SDN architecture, with its protocol being able to manage network switches where packets are distributed. Overall, these processes allow the independent programming of the entire network on individual switches and data centers. Subsequently, the SDN controller is capable of gathering flow status from every switch and conveniently managing its flows. For example, the controller is able to arrange call data packets distributed by OpenFlow switches.

3) SECURITY CHALLENGES IN SDN

Despite an attempt in the SDN architecture to carry security prone in the network management, segregation of the control plane from the data plane creates another type of security risk to the SDN architecture that may be present in particular in three layers: infrastructure, implementation, and control layers. Furthermore, this security prone could result in changes in data, blocking access to the network, DoS, and data breach [17]. A large number of attacks take place after the introduction of centralized control by the SDN architecture. In [18], the chance of an attack occurring on the SDN controller was illustrated. Upon the commissioning of the controller, the attacker is able to change device rules and to reject any access by legitimate users to existing resources (DoS attack). Rather than DoS attacks being the sole attacks for SDN, there are other common attacks, including vulnerability scan, port proves, side-channel, and man-in-the-middle (MITM) [18], [19].

The incorporation of an IDS into SDN architecture is among the ideal methods to develop a secure SDN environment [20], [21]. In this case, IDS is a system made for the detection and notification of unpermitted attempts at access, adjustments, and/or restrictions of computer system resources. The system commonly identifies harmful traffic

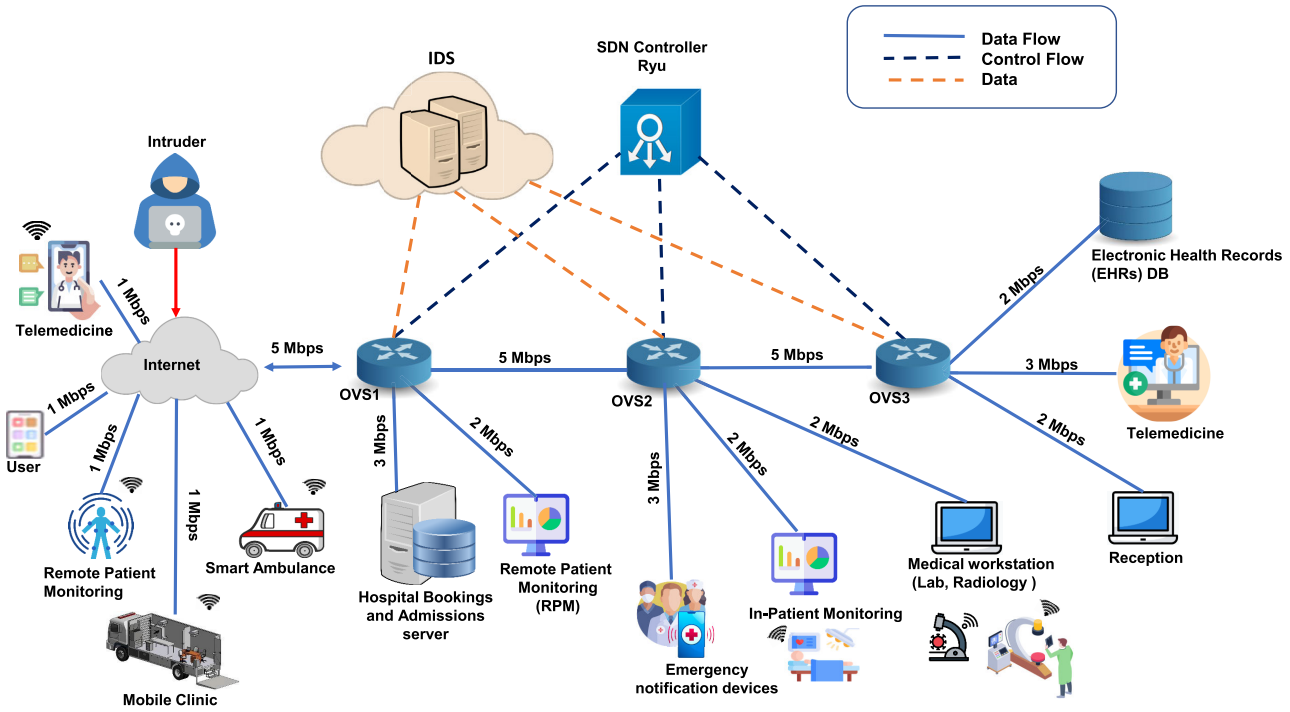

FIGURE 2. Communication topology utilizing SDN in the Healthcare sector.

and threats against the network or a single host computer. Generally, two categories of IDS are the most commonly used [22], [23]: host IDS (HIDS) and network IDS (NIDS). To be specific, installation and use of an HIDS is conducted on every system or network as an individual device that monitors coming and outgoing packets in the network or system. It also alerts the user or administrator to any a proven attack or suspicious activities identified in the system. Furthermore, HIDS commonly takes place through a snapshot taken of available files, which is then compared with the previous snapshot of system files to detect unpermitted activities. It is also a system that detects irregular behavior and attacks in the network through examination of network traffic and supervision of various hosts over the network environment. Commonly, an NIDS obtains access to network traffic by connecting to a configured switch, hub, and network tap for port mirroring. Given this, our study aims toward the implementation of an NIDS in the SDN environment.

B. PREVIOUS WORKS

SDN is widely used in basic core industry platforms, such as cloud and medical environments; therefore, intrusion attacks for SDN can easily interfere with a large number of services that are parasitic on the platform, resulting in major network security incidents. To prevent this problem, a large number of IDSs have been proposed by the academic community. IDSs can be categorized as either HIDS or NIDs based on the data source. They can also be categorized based on the detection mechanism as misuse, anomaly, or hybrid IDS. According to the detection method, they can be divided into statistical- or machine learning-based detection schemes [24].

A statistical-based detection scheme is a scheme to distinguish intrusion attacks by extracting statistical features. A threshold is usually set, and traffic exceeding this threshold is judged as an attack. In [25], the authors proposed an entropy-based detection method, by deploying a data collector on the switch side and calculating the entropy value of the stream according to the destination IP address. This scheme relies on the flow defined by the quintuple (source IP, destination IP, source port, destination port, protocol). This detection scheme is fast and efficient, but needs to separately deploy an additional collector, as OpenFlow comes with a flow table that may not specify a specific element, such as the destination IP, in which case the detection scheme is invalid.

A similar algorithm was chosen in [26], but the entropy value was calculated every 50 packets. This method is not reliable in a real-case scenario because, when the traffic of normal data packets is also large, it often leads to misjudgment.

In [27], the authors scored each data packet, and the scoring principle was based on whether the source IP was in the flow table, whether there was a successful TCP connection, the protocol type, and the rate of data communication. The dynamic adaptive threshold algorithm is used to determine whether the attack occurs.

The authors of [28] used packet-in information as the signal of the attack. When the switch encounters a packet that cannot be matched, it triggers the table-miss option, which sends a packet-in message to the controller. When an

attack occurs, this table-miss situation proliferates because of the random source IP technique, resulting in a large number of packet-in messages. However, in a real environment, the controller can very easily be attacked by instantaneous large traffic before it has enough time to respond, as this packet-in information is sent to the controller through the southbound interface. In addition, for HTTP and SSL flooding attacks using a real source IP, it is difficult for the detector to trigger an alarm as a large amount of packet-in information is not activated.

As for the detection scheme based on machine learning, it works by extracting characteristics of the traffic flowing through the switch, and using machine learning methods to identify malicious traffic. In [29], the authors adopted SVM as their classification algorithm, but the attack traffic they generated was only 25 packets per second. This can be correctly detected in a simple experimental setting, but in practical situations, the credibility of the results is still worth checking. Both statistics-based and machine learning-based detection schemes cannot avoid the extraction of traffic characteristics in the OpenFlow environment. Mainstream feature extraction methods are divided into packet-in feature extraction based on southbound channels and feature extraction based on flow table statistics.

The scheme based on packet-in feature extraction operates as explained here. After the switch triggers the table-miss option, it will send packet-in information to the controller. Common SYN, UDP, and ACK floods often use source IP forgery technology to protect the controlled end, which triggers a large number of packet-in callbacks, based on which corresponding abnormal features can be extracted. This mechanism is adopted in [24], [28], and [30]. In addition, packet-in and flow table overflow attacks against SDNs also have destructive effects, as they essentially need to randomly generate matching fields to trigger packet-in. In these attack scenarios, if the solution based on flow table statistics is adopted, the response time is reduced due to congestion of the southbound bandwidth and overflow of the switch flow table.

Schemes based on packet-in feature extraction and on flow table statistics are not suitable for a wide range of attack scenarios. Most researchers focus on improving detection accuracy and ignore the need for quick response, especially when faced with different attack scenarios, so the response time of the system often varies greatly. Some solutions are actually transferred from traditional intrusion detection methods and do not make good use of the global features of the SDN itself. Configuring traditional detection schemes usually requires additional deployment of data collectors, which makes the scheme extremely complex and adds extra overhead.

Works presented in [31] and [32] made use of a Mininet simulation environment. The authors found that, when the number of flow rules in the flow table is too high, the time consumed by acquisition of flow table statistical information greatly increases. Therefore, in the face of new attack methods such as packet-in flooding and flow table overflow, the detection efficiency is greatly reduced.

In [33], the authors discussed a combined approach to the protection of industrial healthcare systems against cyber threats. They proposed an SDN module as the core of their design, combined with a reinforcement learning (RL) algorithm. The SDN module was used to detect and mitigate attacks, while the RL algorithm optimized the system's security level. Through simulations, their approach was able to successfully detect and mitigate cyber threats in industrial healthcare systems.

The authors of [34] studied an approach to the prevention of a type of attack known as a 'hello flood attack' in the Internet of Things (IoT). This attack is executed by sending a large number of requests to IoT devices in order to overload them. The article proposes a combination of deep learning and improved rider optimization algorithms to prevent such an attack from occurring.

In [2], the authors discussed a Bayesian-based trust management system to prevent insider attacks in healthcare software-defined networks. The system makes use of Bayesian inference to evaluate the trustworthiness of nodes in the network, using data collected from the nodes, as well as their behaviors. The system then uses this information to identify and respond to potential attacks. The authors also discussed the potential of using machine learning algorithms to further improve the system's accuracy. Finally, they provided an example implementation of the system and discussed both the benefits and drawbacks of using such a system.

The authors of [35] explored the use of machine learning techniques to detect and mitigate distributed denial of service (DDoS) attacks in SDNs. They studied various ML algorithms, such as decision trees, support vector machines (SVM), and artificial neural networks (ANNs). They also discussed the various parameters that should be optimized with a view to improving the performance of their proposed solution. The paper concludes by presenting a case study to demonstrate the effectiveness of the proposed solution.

In [36], a detection scheme for DDoS attacks on IoT devices was proposed, using hybrid ensemble learning and a genetic algorithm (GA). It proposes a combination of supervised and unsupervised learning techniques to detect DDoS attacks in real-time. The authors also examine the feasibility of using GA to improve the accuracy and efficiency of the detection scheme. Their research is intended to help provide comprehensive security for IoT devices.

In [37], the authors developed an efficient cyberattack detection system on the Internet of medical things (IoMT) smart environment, based on deep recurrent neural network (RNN) and machine learning algorithms. The proposed system is designed to detect malicious activity and protect the IoMT from cyberattacks. The authors also discussed the various challenges associated with the development of such a system, and the importance of using ML algorithms for accurate and efficient detection.

In [38], the authors used an SDN for detecting and mitigating cyberattacks in integrated clinical environments. They proposed a novel SDN-based architecture to detect and mitigate various cyberattacks, such as DDoS attacks, malicious code injection, or unauthorized access.

The authors of [39] discussed the development of a new dataset for intrusion detection in SDNs. This dataset consists of network traffic from an SDN environment, which has been modified to include realistic simulated attack scenarios. The dataset can be used to train and evaluate intrusion detection systems. It also includes labels to help researchers measure the performance of their detection models.

Lyapunov-Krasovskii functions (LKFs) and the second-order weight method (SOWM) are used to estimate the error generated by T-S fuzzy networked control systems under DoS attack [40]. Reciprocally convex matrix inequality (RCMI), proper integral inequalities, and the linear convex combination method (LCCM) are used to analyze the T-S fuzzy networked control systems under stochastic cyber-attacks (SCAs) [41].

The authors of [42] worked on the development of an IDS for SDNs. The IDS, ML-IDSDN, is based on machine learning and aims to detect malicious activity on SDNs. The authors explained the theory behind the system, described its design and architecture, and evaluated its performance. Table 1 introduces a summary of these related papers with their available published results.

III. ML BASED CYBERATTACK DETECTOR (MCAD)

In this part, we introduce the main aspects of our work, first by showing the proposed system and method, including the ML algorithm and the datasets. We will also discuss the network topology and metrics used to measure the performance, as well as details of the tools and methodology used for the experiments.

A. PROPOSED SYSTEM AND DESIGN METHODOLOGY

The proposed model consists of five main phases as depicted in Figure 3, namely: (i) proposing a logical network topology, (ii) data gathering, (iii) data preprocessing, (iv) training and testing the ML model, and (v) deployment on the Ryu controller.

In more detail, the proposed model begins with building the topology and then data gathering, which includes different types of attacks and exploitation (i.e., probe attack, exploit virtual network computing (VNC) port 5900 remote view vulnerability, and exploit Samba server vulnerability) and normal samples.

As shown in Figure 4, four network adaptors were configured on an Ubuntu virtual machine, i.e., (I) enp0s9, (ii) enp0s8, (iii) enp0s10, and (iv) enp0s3. Three Open vSwitch (OVS) bridges were created to connect Mininet, Kali Linux, and Metasploitable machine together using Linux routing and internet protocol (IP) forwarding. The controller on the Ubuntu machine can monitor three different networks.

In addition, the feature extractor tool is implemented using the L3 learning switch application along with the Ofctl_rest application, used as a tool to monitor OVS, using inheritance. Ofctl_rest application deploys a server on port 80 to perform get and post requests to collect network statistical features. The output resulting from the feature extractor tool is 27 statistical features.

In phase 3, i.e., the preprocessing phase, four main steps are performed: (i) data cleansing, (ii) feature transformation, (iii) data scaling, and (iv) data shuffling. This is because ML algorithms perform better when numerical input variables generated from the dataset are scaled to a specific standard range. As a dataset usually contains various different features and each one might have a different range of values or units of measure, the Robust Scaler (RS) is used to normalize the features.

Before splitting the dataset, data shuffling is used to mix up all samples and avoid biases to ensure fairness between subsets. Then, the dataset is split into training, validation, and testing sets in a stratified fashion. First, the dataset is split into training and testing sets with ratios 0.9 and 0.1 respectively. The training dataset is further split into training and validation sets with 0.9 and 0.1 proportions respectively. We used the resources shown in Table 2 to conduct our experiments.

The proposed model uses KNN, decision tree (DT), random forest (RF), naïve Bayes (NB), logistic regression (LR), adaptive boosting (adaboost), and xgboost (XGB) for training on the datasets mentioned earlier. These classification algorithms can construct a mapping function between inputs and output by detecting different patterns and minimizing the error as much as possible, based on the algorithm and the relational complexity between inputs and outputs. The performance is measured through the model's accuracy, as well as measuring the throughput as the model targets real-time systems to ensure the overall quality of the system.

Finally, in the fifth phase, We deployed the best model on the Ryu controller to classify incoming traffic into normal and attack. The deployed model has the ability to block attacks, thus improving different network KPIs.

B. DATASET

Normal and attack traffic can flow through the SDN architecture. In this section, we discuss the generated dataset, starting by discussing different types of attacks that can impact the SDN, then tools used to generate this traffic, followed by developing the application which is responsible for collecting aggregated features from the flows. Finally, different data preprocessing techniques were applied to the dataset before splitting it into training, validation, and test sets.

1) TRAFFIC TYPES

The SDN is subjected to different types of attacks targeting different entities on the network, such as DDoS, Probe, Web, User to Root (U2R), and Remote to Local (R2L). DDoS attacks are a type of DoS attack in which several linked devices, often known as botnets, are used to overload a

TABLE 1. Summary of recent related works for IDSs on SDNs.

Source	Aim	Model/ algorithm	Benchmark dataset	Detected attack	Findings	Challenges/ shortcomings
Radoglou-Grammatikis et al (2021) [33]	To introduce an intrusion detection and prevention system (IDPS) to detect/prevent IEC 60 870-5-104 cyberattacks.	Decision Tree Classifier	Simulated dataset using: 1) seven VMs with IEC-TestServer representing the field devices; 2) a VM with Qtester104 playing the role of a human–machine interface; and 3) three VMs equipped with Metasploit, OpenMUC j60870, and Ettercap, representing the cyberattackers.	IEC 60 870-5-104 cyberattacks	Accuracy: 83.14% False positive rate: 1.53% True positive rate: 83.14% F1: 82.58%	Large number of features, which may increase the latency of the overall system.
Srinivas & Manivannan (2020) [34]	To propose a robust model to detect and prevent HELLO flooding attacks using an optimized deep learning approach.	Optimized deep belief network	Simulated environment.	DoS-based Hello flooding attack	It outperforms state-of-the-art IDSs in terms of length of the shortest path, latency, normalized energy	Trust-aware routing may make the security mechanism invalid when the routing protocol of the network is changed. No real attacks were used to assess the proposed IDS. It was not evaluated in terms of major IDS metrics, including accuracy, precision, and recall.
Meng et al. (2018) [2]	To develop a trust-based approach based on Bayesian inference to figure out malicious devices in a healthcare environment.	Bayesian inference	Simulated SDN environment	Betrayal attack	The proposed IDS has the capability of reducing the trust value of malicious devices more quickly than a challenge-based approach.	Sensitivity and validity were not considered. The performance of the proposed approach was not assessed in an even larger environment.
Sanjeetha et al. (2022) [35]	To propose a model that calculates the threshold limit for the type of application sending data to a particular switch, in real-time using a machine learning model, and determines whether that application traffic is DDoS traffic.	Random decision forests and multi-linear regression.	Kaggle dataset	DDoS, HTTP, FTP, UDP attacks	Proposed IDS is not well assessed against state-of-the-art models. No clear findings were provided	It was not evaluated in terms of major IDS metrics, including accuracy, precision, and recall.
Erfan (2021) [36]	To propose an efficient IDS based on an ensemble machine learning model.	C4.5 decision tree, deep neural network (DNN) and K-KNN.	KDDCup99	DDoS	It outperforms other IDSs in terms of accuracy and false alarm rate.	The benchmark dataset is outdated, and does not relate to SDN/IoT. Furthermore, several new types of attacks are not covered in the dataset used.
Saheed & Arowolo (2021) [37]	To propose an IDS based on a deep recurrent neural network (DRNN) and supervised ML models (random forest, decision tree, KNN, and ridge classifier) in the IoMT environment to classify and forecast unexpected cyberthreats.	DRNN, random forest, decision tree, KNN, and ridge classifier.	NSL KDD	DDoS	Accuracy: 96.08, recall: 85.63, precision: 85.63	The benchmark dataset is outdated, and does not relate to SDN/IoT. Furthermore, several new types of attacks are not covered in the dataset.
Huertas Celdrán (2021) [38]	To propose an SDN/NFV-based architecture for mobile edge computing infrastructure to detect and mitigate cybersecurity attacks exploiting SDN vulnerabilities of ICE in real-time and on-demand.	Network function virtualization	Simulation scenarios	MiTM, ARP cache poisoning, and phantom storm	NA	Proposed IDS is not fully automated since they are using hard-coded rules to detect and mitigate the attacks, instead of intelligent components (ML or DL-based) which are able to detect new cyberattacks affecting not only the SDN plane but also medical devices making up the ICE scenario. In addition, the proposed IDS should be evaluated with different types of attacks to validate its feasibility.
Elsayed et al. (2020) [39]	To generate an attack-specific SDN dataset; publicly available to researchers	Mininet, Kali Linux and Metasploitable 2 server for topology and decision tree, random forest, adaptive boosting learner, k-nearest neighbor, naive bayes, linear kernel and a radial basis function kernel for classification	Simulated dataset	DoS, DDoS, R2L, Web attack, Malware, Probe, U2R	ACC = up to 99.99% PR = up to 99.40% CR = up to 99.90% F1 = up to 99.99% for InSDN dataset	Complexity: 24-50 features were used, causing training time to reach 1820 sec.
Alzahrani & Alenazi (2022) [42]	To build an ML model to classify DDoS and probe attacks based on a generated dataset	Mininet and Ryu controller for topology and decision tree for classification	Simulated dataset	Fin flood, UDP flood, ICMP flood, OSprobe scan, port probe scan, TCP bandwidth flood, and TCP syn flood	F1-score on attack class of 0.9995, F1-score on the normal class of 0.9983, and throughput score of 6,737,147.275 samples per second with a total number of three features	DDoS and Probe attacks are the only attacks covered

TABLE 2. Simulation environment.

Parameter	Value
Virtual machines	Kali Linux VM 2022.2 Metasploitable2 Server VM Ubuntu 8.04 Ubuntu VM 20.04.3
Tools	Mininet 2.3.0, DVWA 1.0.7
Memory	32.0 GB of RAM
CPU	Intel core i7-8565U @ 1.80GHz
Link Bandwidth	1,2,5 Mbps
SDN Framework	Ryu
Open vSwitch	2.13.3
Python	3.8.10

specific target [43]. These numerous requests consume target resources in the form of empty packets, eventually causing the system to collapse. Aside from its harmful effect on the target, it can have an impact on the entire network's resources by reducing the flow of regular network traffic. TCP, UDP, ICMP, DNS, and VoIP flood are examples of DDoS attacks. The focus of this work regarding DDoS attacks is on TCP and UDP flood attacks, generating them by the Scapy tool. Scapy is a reliable interactive packet manipulation program that supports Python. It can generate or decode packets from a variety of protocols, including TCP and UDP, transmit them over the wire, and collect them.

A probe attack is a network sniffing operation that impacts one or a limited number of victims on the network. The attacker can scan a range of working IP addresses, open ports, services, and OS versions of the victims [44]. Sniffed data can be used later to perform more destructive attacks. Nmap was used to perform this kind of attack.

SQL injection is a Web security vulnerability that allows an attacker to manipulate the queries generated by an application to its database based on the client interaction. In general, it enables an attacker to examine data that is not accessible in the happy scenario. This might contain data belonging to other users or any other data that the program can access but regular users cannot. Consequently, an attacker can change the application's content or behavior by modifying, truncating, or dropping tables [45]. The Damn vulnerable Web application (DVWA) is an extremely vulnerable PHP/MySQL Web application. Its primary goal is to assist security professionals in testing their skills and tools in a legal environment for a better understanding of the processes of securing Web applications. It contains the most common online vulnerabilities at varying levels of difficulty using an intuitive web interface. DVWA has been used to perform SQL injection attacks. However, we noticed that this process takes time when performed manually. In order to accelerate this process, an automation script was developed using Python and Selenium to perform this kind of attack.

R2L refers to unauthorized access from a remote source to a local host, such as brute force password guessing and

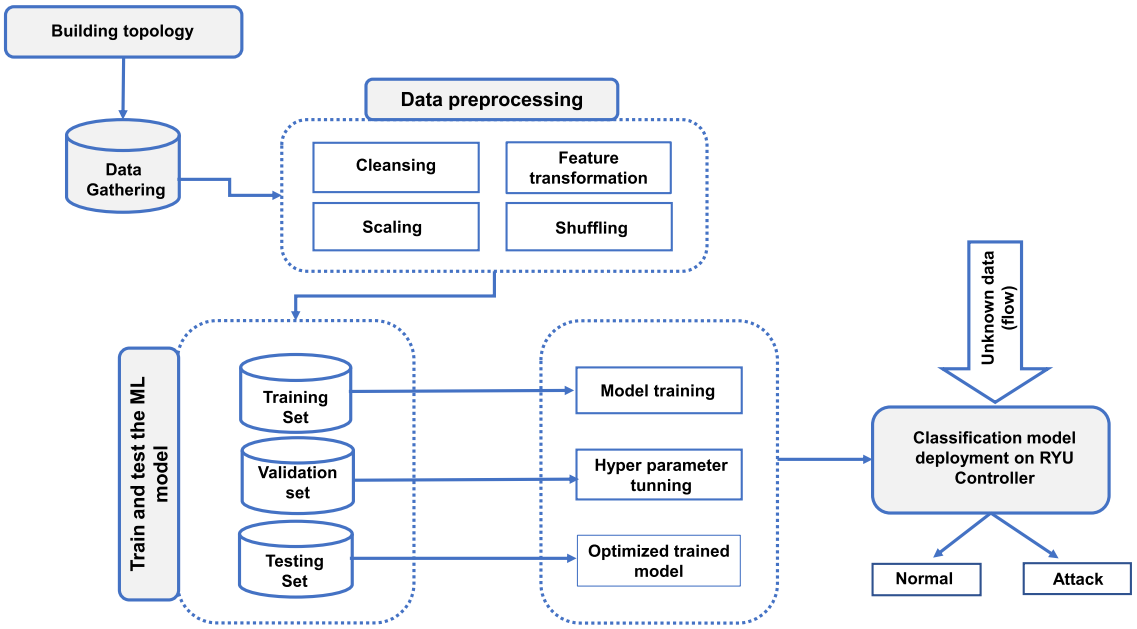

FIGURE 3. Procedural steps of proposed model.

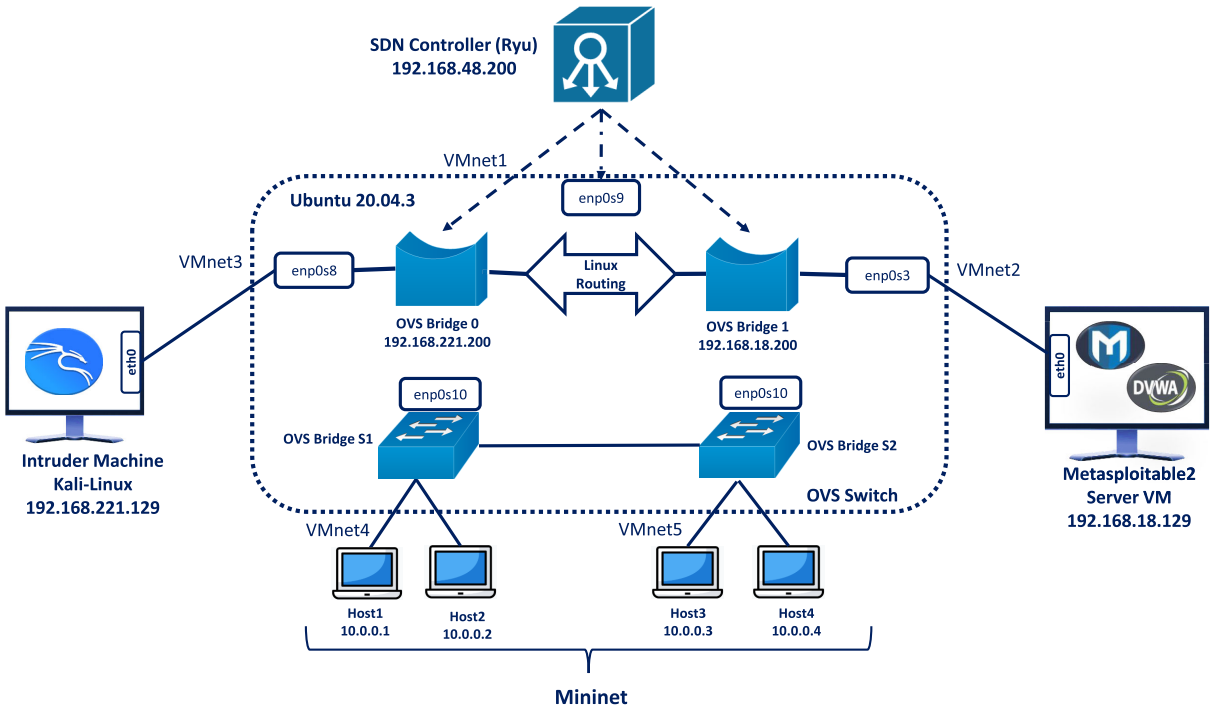

FIGURE 4. Logical network topology.

command injection attacks. A brute force attack uses trial and error to estimate login credentials and encryption keys [46]. Intruders try every available combination with the goal of making an accurate guess. Cracking a password might take a few seconds to several years, depending on its size and complexity. A cyberattack involving the execution of unauthorized commands on a server operating system is known as command injection (CMD). Typically, the threat actor injects commands by taking advantage of an application vulnerability, such as improper input validation [47]. Brute force and CMD attacks were also automated using selenium and performed on DVWA.

Another type of unauthorized access is U2R, which involves exploiting the VNC port 5900 and the Samba

server [48]. Using the Samba server vulnerability, attackers potentially execute arbitrary commands with no authentication requested. An attacker with network access to vulnerable devices and an open port might exploit the VNC port 5900 vulnerability. No privileges or user involvement are required for successful exploitation, which allows the execution of programs with privileged access on the system by sending specially crafted network requests to port 5900/TCP. An attacker might use this exploit to compromise the VNC server's confidentiality, integrity, and availability. The Metasploit framework has been used to implement these attacks on metasploitable2 VM. Table 3 shows the generated attack classes' impact on different locations on SDN, with the tools used for each attack, intruder, and victim.

Iperf, W3m, and D-ITG have been used to generate normal samples, with Iperf's parameter being randomized periodically to pick up a protocol between UDP and TCP, with random port, duration, and bandwidth ranging from 100 Kbps to 10 Mbps. W3m is a text-based web browser that can be accessed from the terminal. We developed a script that performs internet browsing on 1000 different websites, such as Facebook and YouTube, using Python and W3m. D-ITG is a tool that generates TCP or UDP traffic at the packet level, in addition to measuring different network KPIs, such as delay, jitter, and throughput.

2) DATASET COLLECTED

We developed a Layer 3 learning switch application that maintains the switching functionalities of the network. This application was integrated with the ofctl_rest application using inheritance on the developed code. The ofctl_rest application was used to collect the features of normal and attack traffic in addition to blocking the intruder based on the machine learning (ML) classifier decision. We collected 27 different features for each flow with one-second intervals. Table 4 shows the extracted features and their description. in_port and dl_dst were used as decision features to block a certain flow, while other features were used to train the ML classifiers.

3) DATA PREPROCESSING

Data preprocessing is an essential and mandatory phase of machine learning, due to its positive impact on improving overall performance with respect to throughput and accuracy.

We removed identity features; that is to say, src, dst, table_id, in_port, and dl_dst. This step was taken to ensure that the model only focuses on the traffic pattern and is independent of the network identity parameters. After that, features that have zero variance and do not change over samples were removed, that is to say port_rx_dropped, port_tx_dropped, port_rx_errors, port_tx_errors, port_rx_frame_err, port_rx_over_err, port_rx_crc_err and port_collisions. Zero variance features do not have any impact on the model as they do not change at all. This introduces redundancy when used in the training phase.

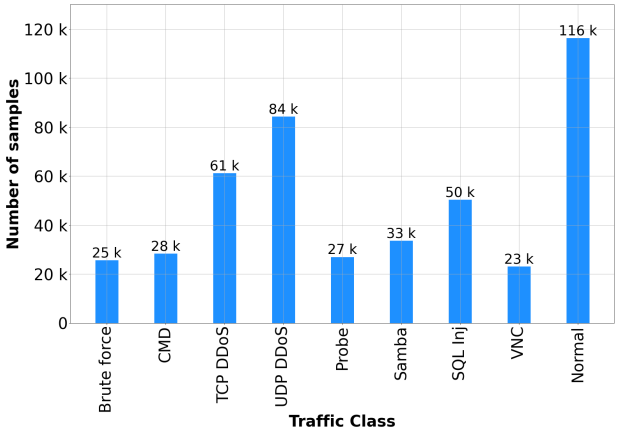

FIGURE 5. Dataset classes.

Division transformation was one of the remaining processes to extract new complex features from existing raw ones. Table 5 shows 13 different transformed features. Raw features that were used to calculate complex features were then dropped to avoid redundancy in the features.

The dataset contains a total of 450,626 samples representing brute force, CMD injection, TCP DDoS, UDP DDoS, probe, Samba, SQL injection, VNC exploitation, and normal traffic. Figure 5 shows the distribution of the classes in the generated dataset. The dataset was then split into training and test sets, with 0.9 and 0.1 ratios respectively, using a stratified approach to preserve the distribution of classes over different sets. The training dataset was further split into training and validation sets, again with 0.9 and 0.1 proportions, respectively. Table 6 shows the distribution of the generated classes over the training, validation, and testing sets

Scaling the input features is mandatory in the machine learning pipeline in order to enhance the performance of the ML algorithms. As a dataset usually contains various different features and each one can have a different range of values or units of measure, RS was used to scale up the dataset inputs [49]. RS transforms the feature vector: each value is subtracted from the feature median and is then divided by the interquartile range (IQR), which is the difference between the 75th percentile and 25th percentile using the formula:

$$new\_value = \frac{current\_value - median}{IQR} \tag{1}$$

Finally, principal component analysis (PCA) was used to reduce the dimensionality of the dataset with respect to the number of features from thirteen to five features, with an explained variance of 0.99. This is achieved by orthogonal linear transformation using Eigen values and vectors of the covariance matrix. Consequently, the data can be mapped into a new coordinate system in which the majority variance can be expressed with fewer dimensions than originally [50].

C. DATASET AVAILABILITY

The MCAD-SDN dataset is publicly available at kaggle.

TABLE 3. Generated attack classes and their associated impacts on SDN elements.

Classes of Attack	Impacted location on SDN	Attack Type	Tool(s)	Intruder Machine	Victim
DDoS	•Application Plane •Controller •Data Plane	TCP Flood, UDP Flood.	Scapy	Mininet Host:10.0.0.2 Mininet Host:10.0.0.3	Mininet Host:10.0.0.1
Probe	•Application Plane •Controller •Data Plane	Network discovery (scan IP Address, service name, operating system and port)	Nmap	Kali Linux VM:192.168.221.129	Metasploitable 2 server VM 192.168.18.129
Web Attacks	•Application Plane	SQL Injection	Selenium	Kali Linux VM: 192.168.221.129	Metasploitable 2 server VM Web server (DVWA): 192.168.18.129
R2L	•Application Plane •Controller •Data Plane	Brute Force Attack, CMD	Selenium	Kali Linux VM: 192.168.221.129	Metasploitable 2 server VM Web server (DVWA): 192.168.18.129
U2R	•Application Plane •Controller •Data Plane	Exploit VNC port 5900, Samba Server	Metasploit framework (mfsconsole)	Kali Linux VM: 192.168.221.129	Metasploitable 2 server VM 192.168.18.129

TABLE 4. Extracted features from ofctl_rest application with their description.

No.	Feature name	Description
1	src	source IP address
2	dst	destination IP address
3	table_id	ID of the table
4	ip_bytes	number of bytes in flow on the IP address level
5	ip_packet	number of packets in flow on the IP address level
6	ip_duration	number of seconds that the flow has been alive on the IP address level
7	in_port	the receiver port number
8	dl_dst	MAC address of the receiver host
9	port_bytes	number of bytes in flow on the port level
10	port_packet	number of packets in flow on the IP address level
11	port_flow_count	number of flows on the port level
12	table_active_count	number of active entries
13	table_lookup_count	number of packets looked up in table
14	table_matched_count	number of packets that hit table
15	port_rx_packets	number of received packets on the port level
16	port_tx_packets	number of transmitted packets on the port level
17	port_rx_bytes	number of received bytes on the port level
18	port_tx_bytes	number of transmitted bytes on the port level
19	port_rx_dropped	number of packets dropped by receiver on the port level
20	port_tx_dropped	number of packets dropped by transmitter on the port level
21	port_rx_errors	number of received errors
22	port_tx_errors	number of transmitted errors
23	port_rx_frame_err	number of frame alignment errors
24	port_rx_over_err	number of received packets with overrun
25	port_rx_crc_err	number of CRC errors
26	port_collisions	number of collisions
27	port_duration_sec	number of seconds that the flow has been alive on the port level

D. CLASSIFICATION ALGORITHMS

For a comprehensive look and to achieve the best possible performance, seven different classification algorithms were used while training and testing the proposed model.

1) K NEAREST NEIGHBOR

In classification and regression tasks, KNN can be used. This method is capable of classifying the incoming sample by matching it to k, the number of samples in the training set, where k is the number of neighbours in the training set which are the closest samples to the input sample. This method considers that an incoming sample belongs to a specific class if the majority of its neighbours also do [51]. Different similarity metrics can be used, but Minkowski distance with $p = 2$, commonly known as the Euclidean distance, is the most commonly used, using the

TABLE 5. Division transformed features.

No.	Feature name	Equation
1	ip_bytes_sec	$\frac{ip\_bytes}{ip\_duration}$
2	ip_packets_sec	$\frac{ip\_packet}{ip\_duration}$
3	ip_bytes_packet	$\frac{ip\_bytes}{ip\_packet}$
4	port_bytes_sec	$\frac{port\_bytes}{ip\_duration}$
5	port_packet_sec	$\frac{port\_packet}{ip\_duration}$
6	port_byte_packet	$\frac{port\_bytes}{port\_packet}$
7	port_flow_count_sec	$\frac{port\_flow\_count}{ip\_duration}$
8	table_matched_lookup	$\frac{table\_matched\_count}{table\_lookup\_count}$
9	table_active_lookup	$\frac{table\_active\_count}{table\_lookup\_count}$
10	port_rx_packets_sec	$\frac{port\_rx\_packets}{port\_duration\_sec}$
11	port_tx_packets_sec	$\frac{port\_tx\_packets}{port\_duration\_sec}$
12	port_rx_bytes_sec	$\frac{port\_rx\_bytes}{port\_duration\_sec}$
13	port_tx_bytes_sec	$\frac{port\_tx\_bytes}{port\_duration\_sec}$

TABLE 6. Distribution of classes over training, validation, and testing sets.

Traffic type	Training	Validation	Test
Brute force	20835	2315	2572
CMD	23061	2562	2847
TCP DDoS	49601	5512	6124
UDP DDoS	68336	7593	8437
Probe	21934	2437	2708
Samba	27270	3030	3366
SQL Inj	40869	4541	5046
VNC	18752	2084	2315
Normal	94348	10483	11648

formula:

$$Distance = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \tag{2}$$

where x and y are two vectors.

2) DECISION TREE

The DT algorithm is a non-parametric supervised learning algorithm used for both classification and regression tasks. It has a hierarchical, tree-like structure consisting of a root node, branches, internal nodes, and leaf nodes. It uses entropy as the basis to make the decision (entropy values can range from 0 to 1). If all samples in dataset S belong to one class, then entropy is zero. The information gain represents the difference in entropy before and after a split on a given attribute. The attribute with the highest information gain produces the best split, because it best classifies the training data according to its target classification. The information gain is usually represented with the following formula:

$$Information\ Gain\ (S, a) = Entropy(S) - \sum_{v} \frac{|S_v|}{|S|}\ Entropy(S_v) \tag{3}$$

where (a) represents a specific attribute or class designation, Entropy(S) is the entropy of dataset S, $|S_v|/|S|$ represents the number of values in S_v in relation to the number of values in dataset S, Entropy S_v is the entropy of the dataset S_v [52].

3) RANDOM FOREST

The RF consists of a large number of these decision trees, which work together as a so-called ensemble. Each individual decision tree gives a prediction, such as a classification result, and the forest uses the result which is supported by most decision trees.

The secret behind the random forest is what is called the principle of the wisdom of many. The basic statement behind this is that many decisions are always better than those of a single individual or even a single decision tree. This concept was first recognized when estimating a continuous set [53].

4) NAIVE BAYES

NB gives a probability for each class that the observation $(x_1, \ldots, x_n)$ belongs to this class K_i, assuming the features are independent. Expressed mathematically: $P(K_i|x_1, \ldots, x_n)$, Bayes' theorem is the backbone of the Naive Bayes algorithm. Bayes' theorem deals with conditional probabilities, i.e., the probability of an event A given that an event B has occurred. One then writes the probability as $P(A|B)$.

Bayes' theorem says:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \tag{4}$$

Although it is called 'naive', the algorithm is not simple; it just uses the "naive" assumption of conditional independence. This means that, if we have the equation $P(K_i|x) = P(x|K_i) * P(K_i))/P(x)$, the numerator on the right corresponds to the joint distribution, i.e., $P(K_i, x_1, \ldots, x_n)$. This is where conditional independence comes in. We assume that x_j and x_k are independent given K_i. This means that the two events do not affect each other. Mathematically, it is expressed in such a way that the probability that event x_j occurs does not change when event x_k occurs, under the condition that $K_i$is given. This is why it is called conditional independence [54].

5) LOGISTIC REGRESSION

LR is a statistical model that uses the sigmoid function, as an equation between x and y.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

As expected, this function only returns values between 0 and 1 for the dependent variable, regardless the values of the independent variable. This is how logistic regression estimates the value of the dependent variable.

In many cases, multiple explanatory variables affect the value of the dependent variable. To model such input datasets, logistic regression formulas assume a linear relationship between various independent variables. You can modify the sigmoid function and calculate the final output variable as follows:

$$y = f(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_n x_n) \quad (6)$$

The symbol β stands for the regression coefficient. The logistic model can assume these coefficient values when given a large enough experimental dataset with known values of both dependent and independent variables [55].

6) AdaBoost

AdaBoost is said to be adaptive because it uses multiple iterations to create a consistently strong learner, by iteratively adding weak learners in a sequenced manner. At each training run, a new weak learner is added to the ensemble and a weight vector is set that focuses on examples that were misclassified in the previous run. The result is a classifier that has a better accuracy than weak learners' classifiers [56].

7) EXTREME GRADIENT BOOSTING

XGBoost is a supervised machine learning method used for classification and regression. This method is based on decision trees, and is an improvement over other methods, such as random forest and gradient enhancement. It works well with large, complicated datasets as it uses different optimization methods.

To fit a training dataset with XGBoost, an initial prediction is made. Residuals are calculated based on the predicted and observed values. A residual similarity value is used to build a decision tree containing these residuals. The similarity of the data in one sheet is calculated, as well as the amplification of the similarity in the subsequent part. Gains are compared to determine a feature and threshold for a node. The output value for each leaf is also calculated using the residuals. Classification typically calculates values using the logarithm of odds and probabilities. The output of the tree becomes the new residual of the dataset, used to build another tree. This process is repeated until the residuals stop decreasing, or a specified number of iterations is reached. Each succeeding tree learns from the previous trees, in contrast to the way random forests are not assigned equal weight. To use this model for predictions, the output of each tree is multiplied by a learning rate and added to the initial prediction to get a final score or classification [57].

E. EVALUATION METRICS

As we propose using ML as the main approach for attack detection, the main metrics to evaluate performance would be accuracy, precision (P), recall (R), and the F1-score.

TABLE 7. Mathematical equations of evaluation metrics.

Metric	Equation
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision of normal class	$\frac{TN}{TN+FN}$
Precision of malignant class	$\frac{TP}{TP+FP}$
Recall of normal class	$\frac{TN}{TN+FP}$
Recall of malignant class	$\frac{TP}{TP+FN}$
F1-score	$\frac{2*P*R}{P+R}$

All these metrics depend on measurements of the true positives (TP), which represent the number of positive samples that are correctly predicted; false negatives (FN) which represent the number of positive samples that are misclassified as negative by the model; true negatives (TN), which represent the number of negative samples that are correctly predicted; and false positives (FP), which represent the number of negative samples that are misclassified as positive by the model. Table 7 summarizes the equations of the different metrics used in this work.

IV. RESULTS AND DISCUSSION

This section shows the results of the trained machine learning models on different performance metrics, followed by the deployment of the best model to show its ability to mitigate the impact of attacks on different network KPIs.

A. SIMULATION RESULTS

In this subsection, we discuss the simulation results for different algorithms against different attacks. Looking at Tables 8 and 9, it is clear that the different approaches have a huge variance in performance, from zero to 1 in precision, recall, and F1 score. As mentioned, we are showing here the performance against eight different attacks, starting from Brute Force and ending with VNC. It is clear that overall score metrics are significantly high for all attacks, especially for TCP DDoS/UDP DDoS classes and normal classes for the KNN, DT, RF classifiers. The performance is relatively lower in the case of the XGB classifier, while the performance is very poor for other classifiers (NB, LR, and AdaBoost).

Furthermore, the general performance in accuracy metrics is the best in the case of random forest, and the worst when using logistic regression. It is clear that the performance of XGBoost is significantly better that the adaptive boosting classifier. Looking at the throughput, it is the highest for decision trees classifiers, a little lower in the case of logistic regression, drops to third place for Naive Bayes, and drops significantly for other classifiers.

TABLE 8. Simulation accuracy results of machine learning algorithms.

Attack name	Metrics	Algorithm						
		KNN	DT	RF	NB	LR	adaboost	XGB
1: Brute force	Precision	0.9645	0.9716	0.9871	0.2345	0.0000	0.3148	0.9180
	Recall	0.9603	0.9708	0.9817	0.2656	0.0000	0.0066	0.7317
	F1-score	0.9624	0.9712	0.9844	0.2491	0.0000	0.0129	0.8144
2: CMD	Precision	0.9653	0.9673	0.9900	0.2500	0.0000	0.0749	0.9677
	Recall	0.9480	0.9677	0.9719	0.0151	0.0000	0.0885	0.8830
	F1-score	0.9566	0.9675	0.9809	0.0285	0.0000	0.0812	0.9234
3: TCP DDOS	Precision	0.9995	1.0000	0.9998	0.0000	0.0000	0.5873	0.9997
	Recall	0.9982	0.9995	0.9998	0.0000	0.0000	0.9944	0.9995
	F1-score	0.9989	0.9998	0.9998	0.0000	0.0000	0.7385	0.9996
4:UDP DDOS	Precision	0.9987	0.9998	0.9998	0.2178	0.2012	0.0000	0.9991
	Recall	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	0.9999
	F1-score	0.9993	0.9999	0.9999	0.3577	0.3350	0.0000	0.9995
5:Probe (OS Port Scan)	Precision	0.9821	0.9937	0.9959	0.0000	0.0000	0.0708	0.9709
	Recall	0.9930	0.9930	0.9982	0.0000	0.0000	0.0114	0.9967
	F1-score	0.9875	0.9934	0.9970	0.0000	0.0000	0.0197	0.9836
6: Samba	Precision	0.9988	0.9991	1.0000	0.0000	0.0000	0.8996	0.9988
	Recall	0.9967	0.9979	0.9982	0.0000	0.0000	0.9851	0.9952
	F1-score	0.9978	0.9985	0.9991	0.0000	0.0000	0.9404	0.9970
7: SQL Inj	Precision	0.9664	0.9792	0.9828	0.1287	1.0000	0.3310	0.8632
	Recall	0.9752	0.9796	0.9948	0.0026	0.0001	0.9443	0.9867
	F1-score	0.9708	0.9794	0.9888	0.0051	0.0002	0.4902	0.9208
8: VNC	Precision	0.9952	0.9944	0.9974	0.0328	0.0000	0.0000	0.9926
	Recall	0.9927	0.9970	0.9987	0.0009	0.0000	0.0000	0.9879
	F1-score	0.9939	0.9957	0.9981	0.0017	0.0000	0.0000	0.9903
Normal	Precision	1.0000	0.9998	0.9999	0.9971	1.0000	0.6666	0.9999
	Recall	0.9997	0.9998	0.9998	0.2642	0.2684	0.7301	0.9994
	F1-score	0.9998	0.9998	0.9999	0.4177	0.4232	0.6969	0.9997

TABLE 9. Simulation complexity results of machine learning algorithms.

Algorithm	Training time (sec)	Throughput (samples per sec)
KNN	0.414	39977
DT	2.3480	5709692
RF	67.429	84723
NB	0.1105	1776875
LR	31.202	4477375
adaboost	22.073	61278
XGB	100.95	52103

The decision tree was chosen as the best model to fit a real-time system, because it outperformed other proposed models in throughput, as shown in Figure 6. Although the random forest model outperformed all other models in accuracy metrics, including F1-score, as shown in Figure 7, the difference from the decision tree was not great, with a delta of 0.0001 on the normal class and 0.0053 on attack classes.

MCAD was compared with the latest research on developing IDSs. MCAD outperformed IDSs proposed on [33] and [37] on attacks (A) and normal (N) traffic, as shown in Table 10. Although IDSs proposed in [39] and [42] have slightly higher accuracy performance than MCAD (Figure 8), they are lacking a small number of features and/or diversity of attack types. In [39], the authors used 48 features, compared to MCAD, which uses five features only: this reflects a more complex model than MCAD. In [42], the authors proposed a model that works only against DDoS and probe attacks, compared to MCAD, which works against a wide spectrum of attacks.

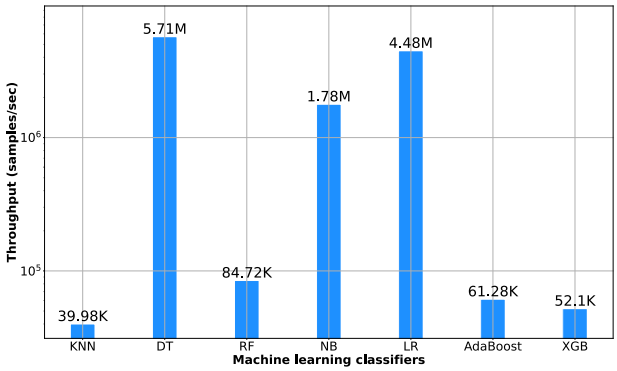

FIGURE 6. Throughput of machine learning algorithms.

B. DEPLOYMENT OF THE MCAD MODEL

In this section, we show the impact of the MCAD model on improving the different network KPIs. We conducted three

TABLE 10. Comparison between IDSs and MCAD.

Source	AC	P(A)	P(N)	R(A)	R(N)	F(A)	F(N)
Radoglou-Grammatikis et al (2021) [33]	0.8318	N/A	N/A	0.8314	0.9847	N/A	N/A
Saheed & Arowolo (2021) [37]	0.9608	0.8563	N/A	0.8563	N/A	0.8563	N/A
Elsayed et al (2020) [39]	N/A	0.994115	N/A	0.997168	N/A	0.99564	N/A
Alzahrani & Alenazi (2022) [42]	0.9992	0.9992	0.999	0.9996	0.9976	0.9994	0.9983
MCAD	0.9924	0.9901	0.9947	0.99	0.9949	0.9895	0.9948

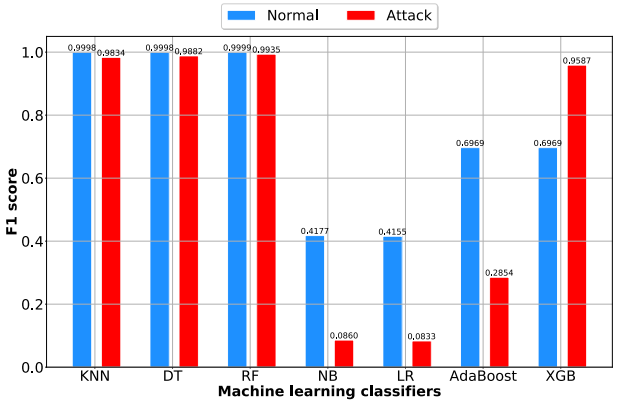

FIGURE 7. F1-score of machine learning algorithms.

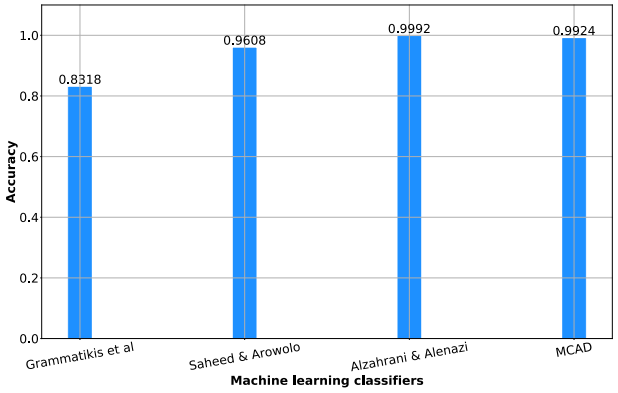

FIGURE 8. Accuracy comparison between IDSs and MCAD.

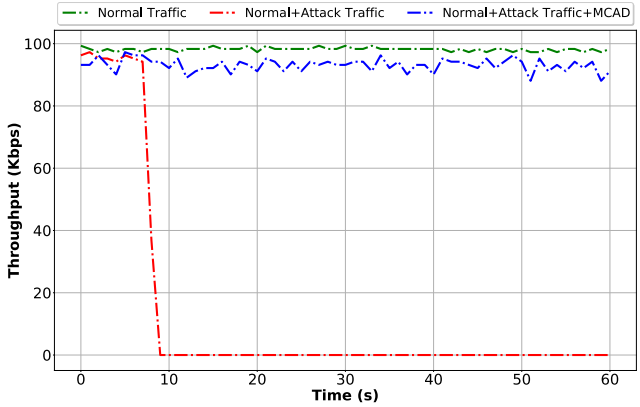

FIGURE 9. Network performance comparison: throughput of the network for normal traffic scenario vs. normal traffic with attacks vs. MCAD.

different practical scenarios while collecting network KPIs through D-ITG. The first scenario is only normal traffic, while the second is normal traffic while the network is under

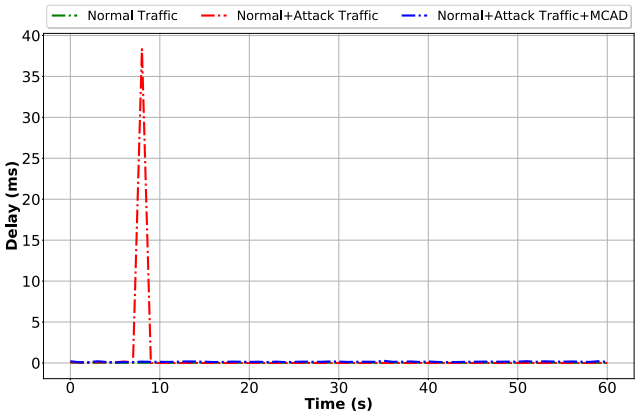

FIGURE 10. Network performance comparison: delay of the network for normal traffic scenario vs. normal traffic with attacks vs. MCAD.

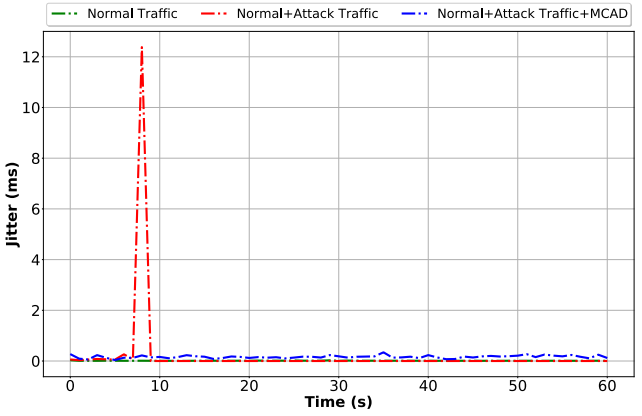

FIGURE 11. Network performance comparison: jitter of the network for normal traffic scenario vs. normal traffic with attacks vs. MCAD.

attack. Finally, in the third scenario, we deployed the MCAD model to classify and block attack flows while the network is under attack. The attack traffic had an adverse impact on throughput, jitter, and delay. The average throughput dropped from 98.19 to 13.13 Kbps, average delay increased from 0.031 to 0.64 ms, and average jitter also increased from 0.011 to 0.214 ms. After the deployment of MCAD, the average network KPIs improved, achieving 93.13 Kbps, 0.149 ms, and 0.164 ms for throughput, delay, and jitter, respectively. Figure 9 represents the throughput of the three scenarios, which increased by 609% after deploying MCAD. Figure 10 and Figure 11 shows the delay and jitter of the three scenarios, which were reduced by 77% and 23%, respectively, after deploying MCAD.

V. CONCLUSION AND FUTURE WORKS

In this paper, we introduced a full investigation of the critical problem related to having a generalized method of detection of attacks and threats in the SDN environment in healthcare systems. We proposed a new detection model, MCAD, which uses machine learning to achieve better and more efficient performance, with a wider spectrum that can cover many different attacks.

To make sure that our model can perform required tasks, we contemplated different attack scenarios which match real-world scenarios, and then analyzed the impact of developed attacks that can be identified by the model on different ML algorithms. It can be seen clearly that the model provides really good performance for most of the attacks with some of the ML algorithms used. Using RF for detection gives outstanding performance with all types of attacks, even that that have patterns that are hard to distinguish from normal traffic. Other techniques, like KNN and DT, provide really close performance, while some other tested ML algorithms struggled to detect some types of attacks.

In the near future, we aim to extend this work in three different directions; first by using other ML techniques, then by considering more attacks/combinations of attacks in the datasets; and finally by testing the models on a more complex network model that may lead us to change the model's architecture.

REFERENCES

[1] M. Jarschel, T. Zinner, T. Hossfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, attributes, and use cases: A compass for SDN," *IEEE Commun. Mag.*, vol. 52, no. 6, pp. 210–217, Jun. 2014.

[2] W. Meng, K.-K.-R. Choo, S. Furnell, A. V. Vasilakos, and C. W. Probst, "Towards Bayesian-based trust management for insider attacks in healthcare software-defined networks," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 2, pp. 761–773, Jun. 2018.

[3] J. T. Kelly, K. L. Campbell, E. Gong, and P. Scuffham, "The Internet of Things: Impact and implications for health care delivery," *J. Med. Internet Res.*, vol. 22, p. 11, Nov. 2020.

[4] (2022). *Networked Medical Devices: Security and Privacy Threats—Symantec—[PDF Document]*. [Online]. Available: https://fdocuments.net/document/networked-medical-devices-security-and-privacy-threats-symantec.html

[5] P. A. Williams and A. J. Woodward, "Cybersecurity vulnerabilities in medical devices: A complex environment and multifaceted problem," *Med. Devices, Evidence Res.*, vol. 8, pp. 305–316, Jul. 2015.

[6] C. M. Williams, R. Chaturvedi, and K. Chakravarthy, "Cybersecurity risks in a pandemic," *J. Med. Internet Res.*, vol. 22, no. 9, Sep. 2020, Art. no. e23692.

[7] N. Thamer and R. Alubady, "A survey of ransomware attacks for healthcare systems: Risks, challenges, solutions and opportunity of research," in *Proc. 1st Babylon Int. Conf. Inf. Technol. Sci. (BICITS)*, I. Babil, Ed., Apr. 2021, pp. 210–216.

[8] H. Babbar, S. Rani, and S. A. AlQahtani, "Intelligent edge load migration in SDN-IIoT for smart healthcare," *IEEE Trans. Ind. Informat.*, vol. 18, no. 11, pp. 8058–8064, Nov. 2022.

[9] R. Hasan, S. Zawoad, S. Noor, M. M. Haque, and D. Burke, "How secure is the healthcare network from insider attacks? An audit guideline for vulnerability analysis," in *Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jun. 2016, pp. 417–422.

[10] (Apr. 2015). *92% of Healthcare IT Admins Fear Insider Threats Thales*. Accessed: Mar. 21, 2023. [Online]. Available: https://cpl.thalesgroup.com/about-us/newsroom/news-releases/92-healthcare-it-admins-fear-insider-threats

[11] D. Chaulagain, K. Pudashine, R. Paudyal, S. Mishra, and S. Shakya, "OpenFlow-based dynamic traffic distribution in software-defined networks," in *Mobile Computing and Sustainable Informatics*. Singapore: Springer, Jul. 2021, pp. 259–272.

[12] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based comparison and selection of software defined networking (SDN) controllers," in *Proc. World Congr. Comput. Appl. Inf. Syst. (WCCAIS)*, Jan. 2014, pp. 1–7.

[13] T. Mekki, I. Jabri, A. Rachedi, and L. Chaari, "Software-defined networking in vehicular networks: A survey," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 10, pp. 1–10, Apr. 2021, doi: 10.1002/ett.4265.

[14] Z. Ghaffar, A. Alshahrani, M. Fayaz, A. M. Alghamdi, and J. Gwak, "A topical review on machine learning, software defined networking, Internet of Things applications: Research limitations and challenges," *Electronics*, vol. 10, no. 8, p. 880, Apr. 2021, doi: 10.3390/electronics10080880.

[15] C.-S. Li and W. Liao, "Software defined networks [guest editorial]," *IEEE Commun. Mag.*, vol. 51, no. 2, p. 113, Feb. 2013.

[16] M. H. Rehmani, A. Davy, B. Jennings, and C. Assi, "Software defined networks-based smart grid communication: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2637–2670, 3rd Quart., 2019.

[17] L. F. Eliyan and R. Di Pietro, "DoS and DDoS attacks in software defined networks: A survey of existing solutions and research challenges," *Future Gener. Comput. Syst.*, vol. 122, pp. 149–171, Sep. 2021, doi: 10.1016/j.future.2021.03.011.

[18] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 151–152, doi: 10.1145/2491185.2491222.

[19] B. Mladenov and G. Iliev, "Studying the effect of internal DOS attacks over SDN controller during switch registration process," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, Jul. 2022, pp. 1–4.

[20] H. Domínguez-Limaico, W. N. Quilca, M. Zambrano, F. Cuzme-Rodríguez, and E. Maya-Olalla, "Intruder detection system based artificial neural network for software defined network," in *Proc. Int. Conf. Technol. Res.* Cham, Switzerland: Springer, Aug. 2022, pp. 315–328.

[21] S. A. Mehdi and S. Z. Hussain, "Survey on intrusion detection system in IoT network," in *Proc. Int. Conf. Innov. Comput. Commun.* Singapore: Springer, Sep. 2022, pp. 721–732.

[22] V. Ponnusamy, M. Humayun, N. Z. Jhanjhi, A. Yichiet, and M. F. Almufareh, "Intrusion detection systems in Internet of Things and mobile ad-hoc networks," *Comput. Syst. Sci. Eng.*, vol. 40, no. 3, pp. 1199–1215, 2022, doi: 10.32604/csse.2022.018518.

[23] K. Malasri and L. Wang, "Securing wireless implantable devices for healthcare: Ideas and challenges," *IEEE Commun. Mag.*, vol. 47, no. 7, pp. 74–80, Jul. 2009.

[24] 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.

[25] R. Wang, Z. Jia, and L. Ju, "An entropy-based distributed DDoS detection mechanism in software-defined networking," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, vol. 1, Aug. 2015, pp. 310–317.

[26] 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.

[27] S. Murtuza and K. Asawa, "Mitigation and detection of DDoS attacks in software defined networks," in *Proc. 11th Int. Conf. Contemp. Comput.*, Aug. 2018, pp. 1–3.

[28] X. You, Y. Feng, and K. Sakurai, "Packet in message based DDoS attack detection in SDN network using OpenFlow," in *Proc. 5th Int. Symp. Comput. Netw. (CANDAR)*, Nov. 2017, pp. 522–528.

[29] S. Y. Mehr and B. Ramamurthy, "An SVM based DDoS attack detection method for Ryu SDN controller," in *Proc. 15th Int. Conf. Emerg. Netw. Exp. Technol.*, New York, NY, USA, Dec. 2019, pp. 72–73, doi: 10.1145/3360468.3368183.

[30] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based DDoS detection system in software-defined networking (SDN)," *ICST Trans. Secur. Saf.*, vol. 4, no. 12, Dec. 2017, Art. no. 153515. [Online]. Available: https://publications.eai.eu/index.php/sesa/article/view/211

[31] G. Lucky, F. Jjunju, and A. Marshall, "A lightweight decision-tree algorithm for detecting DDoS flooding attacks," in *Proc. IEEE 20th Int. Conf. Softw. Quality Rel. Secur. Companion (QRS-C)*, Dec. 2020, pp. 382–389.

[32] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, "A DDoS attack detection method based on SVM in software defined network," *Secur. Commun. Netw.*, vol. 2018, pp. 1–8, Jan. 2018.
[33] P. Radoglou-Grammatikis, K. Rompolos, P. Sarigiannidis, V. Argyriou, T. Lagkas, A. Sarigiannidis, S. Goudos, and S. Wan, "Modeling, detecting, and mitigating threats against industrial healthcare systems: A combined software defined networking and reinforcement learning approach," *IEEE Trans. Ind. Informat.*, vol. 18, no. 3, pp. 2041–2052, Mar. 2022.
[34] T. A. S. Srinivas and S. S. Manivannan, "Prevention of hello flood attack in IoT using combination of deep learning with improved rider optimization algorithm," *Comput. Commun.*, vol. 163, pp. 162–175, Nov. 2020.
[35] A. Kanavalli, A. Gupta, A. Pattanaik, and S. Agarwal, "Real-time DDoS detection and mitigation in software defined networks using machine learning techniques," *Int. J. Comput.*, vol. 10, pp. 353–359, Sep. 2022. [Online]. Available: https://computingonline.net/computing/article/view/2691
[36] A. Erfan, "DDoS attack detection scheme using hybrid ensemble learning and ga algorithm for Internet of Things," *PalArch's J. Archaeol. Egypt/Egyptol.*, vol. 18, no. 18, pp. 521–546, Jan. 2022. [Online]. Available: https://archives.palarch.nl/index.php/jae/article/view/10546
[37] Y. K. Saheed and M. O. Arowolo, "Efficient cyber attack detection on the Internet of Medical Things-smart environment based on deep recurrent neural network and machine learning algorithms," *IEEE Access*, vol. 9, pp. 161546–161554, 2021.
[38] A. H. Celdrán, K. K. Karmakar, F. Gómez Mármol, and V. Varadharajan, "Detecting and mitigating cyberattacks using software defined networks for integrated clinical environments," *Peer-Peer Netw. Appl.*, vol. 14, no. 5, pp. 2719–2734, Sep. 2021.
[39] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "InSDN: A novel SDN intrusion dataset," *IEEE Access*, vol. 8, pp. 165263–165284, 2020.
[40] X. Cai, K. Shi, K. She, S. Zhong, Y. Soh, and Y. Yu, "Performance error estimation and elastic integral event triggering mechanism design for T–S fuzzy networked control system under dos attacks," *IEEE Trans. Fuzzy Syst.*, vol. 31, no. 4, pp. 1–12, Apr. 2023.
[41] X. Cai, K. Shi, K. She, S. Zhong, and Y. Tang, "Quantized sampled-data control tactic for T–S fuzzy NCS under stochastic cyber-attacks and its application to truck-trailer system," *IEEE Trans. Veh. Technol.*, vol. 71, no. 7, pp. 7023–7032, Jul. 2022.
[42] A. O. Alzahrani and M. J. F. Alenazi, "ML-IDSDN: Machine learning based intrusion detection system for software-defined network," *Concurrency Comput., Pract. Exper.*, vol. 35, no. 1, pp. 1–12, Jan. 2023.
[43] K. S. Bhosale, M. Nenova, and G. Iliev, "The distributed denial of service attacks (DDoS) prevention mechanisms on application layer," in *Proc. 13th Int. Conf. Adv. Technol., Syst. Services Telecommun. (TELSIKS)*, Oct. 2017, pp. 136–139.
[44] A. Almazyad, L. Halman, and A. Alsaeed, "Probe attack detection using an improved intrusion detection system," *Comput., Mater. Continua*, vol. 74, no. 3, pp. 4769–4784, 2023, doi: 10.32604/cmc.2023.033382.
[45] A. Sadeghian, M. Zamani, and S. M. Abdullah, "A taxonomy of SQL injection attacks," in *Proc. Int. Conf. Informat. Creative Multimedia*, Sep. 2013, pp. 269–273.
[46] L. Zhang-Kennedy, S. Chiasson, and R. Biddle, "Password advice shouldn't be boring: Visualizing password guessing attacks," in *Proc. APWG eCrime Researchers Summit*, Sep. 2013, pp. 1–11.
[47] Z. Su and G. Wassermann, "The essence of command injection attacks in web applications," *ACM SIGPLAN Notices*, vol. 41, no. 1, pp. 372–382, Jan. 2006.
[48] M. Pivarníková, P. Sokol, and T. Bajtoš, "Early-stage detection of cyber attacks," *Information*, vol. 11, no. 12, p. 560, Nov. 2020.
[49] K. V. A. Reddy, S. R. Ambati, Y. S. R. Reddy, and A. N. Reddy, "AdaBoost for Parkinson's disease detection using robust scaler and SFS from acoustic features," in *Proc. Smart Technol., Commun. Robot. (STCR)*, Oct. 2021, pp. 1–6.
[50] I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Philos. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 374, Apr. 2016, Art. no. 20150202, doi: 10.1098/rsta.2015.0202.
[51] P. Cunningham and S. J. Delany, "K-nearest neighbour classifiers: 2nd edition (with Python examples)," 2020, *arXiv:2004.04523*.
[52] E. H. Sussenguth, "An algorithm for automatic design of logical cryogenic circuits," *IEEE Trans. Electron. Comput.*, vol. EC-10, no. 4, pp. 623–630, Dec. 1961.
[53] P. H. Swain and H. Hauska, "The decision tree classifier: Design and potential," *IEEE Trans. Geosci. Electron.*, vol. GE-15, no. 3, pp. 142–147, Jul. 1977.
[54] Y. Ji, S. Yu, and Y. Zhang, "A novel Naive Bayes model: Packaged hidden Naive Bayes," in *Proc. 6th IEEE Joint Int. Inf. Technol. Artif. Intell. Conf.*, Aug. 2011, pp. 484–487.
[55] X. Zou, Y. Hu, Z. Tian, and K. Shen, "Logistic regression model optimization and case analysis," in *Proc. IEEE 7th Int. Conf. Comput. Sci. Netw. Technol. (ICCSNT)*, Oct. 2019, pp. 135–139.
[56] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, pp. 119–139, Aug. 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002200009791504X
[57] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," 2016, *arXiv:1603.02754*.

LAILA M. HALMAN (Student Member, IEEE) received the B.S. degree (Hons.) in networks and telecommunication systems from Princess Nora Bint Abdul Rahman University, Riyadh, Saudi Arabia, in 2014. She is currently pursuing the M.S. degree with the Department of Computer Engineering, King Saud University, Riyadh. Her research interests include network security, network topology modeling, and machine learning.

MOHAMMED J. F. ALENAZI (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer engineering from the University of Kansas, in 2010, 2012, and 2015, respectively. He is currently an Associate Professor of computer engineering with King Saud University. His research interests include the design, implementation, and analysis of resilient and survivable networks; network routing design and implementation; the development and simulation of network architectures and protocols; the performance evaluation of communication networks; an algorithmic graph approach for modeling networks; and mobile ad hoc network (MANET) routing protocols. He is a member of the ACM.