Received August 5, 2020, accepted August 21, 2020, date of publication August 25, 2020, date of current version September 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3019330

A Flexible SDN-Based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks Using Machine Learning

JESÚS ARTURO PÉREZ-DÍAZ[1], ISMAEL AMEZCUA VALDOVINOS[2], KIM-KWANG RAYMOND CHOO[3,4], (Senior Member, IEEE), AND DAKAI ZHU[4], (Senior Member, IEEE)

[1]Escuela de Ingeniería y Ciencias, Tecnologico de Monterrey, Monterrey 64849, México
[2]Facultad de Telemática, Universidad de Colima, Colima 28040, México
[3]Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249, USA
[4]Department of Computer Science, The University of Texas at San Antonio, San Antonio, TX 78249, USA

Corresponding author: Ismael Amezcua Valdovinos (ismaelamezcua@ucol.mx)

This work has been partially supported by the joint Seed Funding Program from The University of Texas at San Antonio and Tecnologico de Monterrey "UTSA & ITESM seed funding program" and partially supported by the project "Red temática Ciencia y Tecnología para el desarrollo (CYTED) 519RT0580" by the Ibero-American Science and Technology Program for development CYTED.

ABSTRACT While there have been extensive studies of denial of service (DoS) attacks and DDoS attack mitigation, such attacks remain challenging to mitigate. For example, Low-Rate DDoS (LR-DDoS) attacks are known to be difficult to detect, particularly in a software-defined network (SDN). Hence, in this paper we present a flexible modular architecture that allows the identification and mitigation of LR-DDoS attacks in SDN settings. Specifically, we train the intrusion detection system (IDS) in our architecture using six machine learning (ML) models (i.e., J48, Random Tree, REP Tree, Random Forest, Multi-Layer Perceptron (MLP), and Support Vector Machines (SVM)) and evaluate their performance using the Canadian Institute of Cybersecurity (CIC) DoS dataset. The findings from the evaluation demonstrate that our approach achieves a detection rate of 95%, despite the difficulty in detecting LR-DoS attacks. We also remark that in our deployment, we use the open network operating system (ONOS) controller running on Mininet virtual machine in order for our simulated environment to be as close to real-world production networks as possible. In our testing topology, the intrusion prevention detection system mitigates all attacks previously detected by the IDS system. This demonstrates the utility of our architecture in identifying and mitigating LR-DDoS attacks.

INDEX TERMS DDoS attack mitigation, low-rate DDoS (LR-DDoS) attacks, machine learning, software-defined network (SDN).

I. INTRODUCTION

Low-rate denial of service (LR-DDoS) attacks is one of the more challenging denial of service (DoS) attack types to detect, and these attacks are designed to exhaust computing resources on servers. Unlike high-rate distributed DoS (DDoS) attacks, an LR-DDoS attack does not flood the network with high traffic loads. Instead, it carefully triggers specific protocol mechanisms such as TCP's timeout retransmission [1], [2], congestion control [3] mechanisms, and HTTP's keep alive mechanism [4], to deplete the target's computing resources.

DDoS attack detection approaches can be broadly categorized into signature-based and anomaly-based approaches [5], [6]. The former uses the identified patterns or strings from protocol header fields as signatures to match incoming traffic and determine if the flow is malicious (or not). In anomaly-based approaches, a model of normal network traffic is developed and compared with incoming traffic. This allows the classification of normal and malicious (or anomalous) traffic. No system or approach is foolproof. For example, an attacker can fool detection systems to gradually accept malicious traffic as normal [7]. We also observe that most anomaly-based

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

approaches for LR-DDoS detection are based on thresholds [8], and one challenge associated with such an approach is the computation of an optimal value for such parameters.

It is also challenging to deploy effective LR-DDoS attack mitigation solutions, in practice [8]. For example, existing solutions may require updating of the router's firmware, which may not be practical in a number of situations. Given the increasing popularity of software-defined network (SDN) [9], there have been reported LR-DDoS attacks targeting such networks [10], [11]. SDN is a relatively new networking paradigm, designed to facilitate the decoupling of control and forwarding planes from network devices (e.g., routers and switches) and provide a logically centralized control and management entity. However, SDN can also be leveraged to facilitate the detection and mitigation of LR-DDoS attacks. For example, SDN provides a programmable feature, in which network operators can develop and deploy network applications that run on top of the controller to provide network functionality. Such features can potentially be used to facilitate the deployment of detection and mitigation mechanisms for LR-DDoS attacks. Moreover, the controller provides an environment in which a vast number of programming libraries can be used to develop networking applications. In other words, one can utilize state-of-the-art technologies such as machine and deep learning algorithms to enhance the detection and mitigation of LR-DDoS attacks.

In this paper, we introduce a new versatile architecture for LR-DDoS attack detection and mitigation in SDN environments using machine learning techniques. Specifically, this architecture comprises an intrusion prevention system (IPS), which will forward the flows to the intrusion detection system (IDS) API. This will allow us to determine whether the flow is malicious (or not). The IDS API will identify the flow using one of several previously trained machine learning (ML) models. This API is programming language and framework independent, and hence we can use different programming languages and frameworks to implement and train the AI models. Once the IDS API returns the result, the IPS module running on the controller will process the flow accordingly to the mitigation strategy of the architecture if the flow is determined to be an attack. In summary, in this paper:

- We present a flexible security SDN-based architecture aimed at LR-DDoS attack detection and mitigation through the use of multiple machine learning and deep learning techniques.
- We implement and demonstrate the potential of the proposed approach in detecting and mitigating several LR-DDoS attacks, namely: DDoSSim [12], GoldenEye [13], H.U.L.K. [14], R.U.D.Y., Slow Body, Slow Headers, Slowloris, and Slow Read.
- We evaluate the performance of six machine and deep learning techniques for LR-DDoS attacks (i.e., J48, Random Trees, REP Tree, Random Forest, Multi-Layer Perceptron (MLP), and Support Vector Machines (SVM)) in LR-DDoS attack detection and mitigation.

Our modular architecture will allow system implementers to easily replace or enhance a module, API, or ML model without affecting the rest of the architecture. In addition, computationally demanding modules such as the IDS can be located outside the controller. Moreover, the IDS communicates with the controller through an Identification API, which is platform-independent. In other words, system implementers can use any programming language and libraries as needed. Another feature is that our IDS is capable of distinguishing between anomalous and normal traffic flows and determining the type of LR-DDoS attack being carried out.

The rest of the paper is organized as follows. Sections II and III respectively describe LR-DDoS attacks and summarize current LR-DDoS attack detection and mitigation approaches. Section IV describes our proposed architecture. Section V and VI describe the evaluation setup and findings. Specifically, we evaluate the effectiveness of our proposed approach in detecting and mitigating the following LR-DDoS attacks: DDoSSim, GoldenEye, H.U.L.K., R.U.D.Y., Slow Body, Slow Headers, Slowloris, and Slow Read; using six ML techniques (i.e., J48, Random Trees, REP Tree, Random Forest, Multi-Layer Perceptron (MLP), and Support Vector Machines (SVM)). Finally, the last section concludes this paper.

II. PRELIMINARIES

This section briefly describes LR-DDoS attacks, prior to introducing LR-DDoS detection and mitigation techniques in the next section.

A. LOW-RATE DoS ATTACKS

Kuzmanovic and Knightly [1], [15] introduced *shrew attacks*, which are a low-rate attacks targeting TCP's retransmission time-out mechanism in order to deny bandwidth to legitimate TCP flows. Shrew attacks consist of carefully chosen short malicious bursts that repeat at a fixed slow-timescale frequency. The authors explained that the effectiveness of low-rate attacks depends on the ability to create correlated packet losses, forcing TCP to enter into retransmission timeouts.

While LR-DDoS attacks are still an ongoing concern to highly centralized services such as cloud computing and big data service platforms, they are less studied and reported in comparison to DDoS attacks. This is, perhaps, as explained by Wu, *et al.* [16]:

- An attacker must achieve accurate traffic synchronization in their implementations. This complicates the implementation of LR-DDoS attacks, in practice.
- LR-DDoS attacks are 'integrated' with legitimate traffic. Thus, network operators may attribute low performance to system equipment or line failures. In other words, LR-DDoS attacks are not detected and hence, under-reported.
- It is difficult to extract and analyze characteristics of LR-DDoS attacks. To avoid panic among users, network

operators are not reporting such attacks since there is insufficient evidence to classify performance issues as attacks.

LR-DDoS attacks are hard to detect since they have the same characteristics as legitimate traffic and are hidden in background traffic. Attacks are launched through a single attack source and its average rate is low enough, so the number of packets sent is very small and challenging to detect. Thus, common DDoS attack detection mechanisms are not effective in detecting LR-DDoS attacks.

A number of researchers have also demonstrated how the effectiveness of LR-DDoS attacks can be improved. Li, *et al.* [17], for example, proposed a multiplexing technique to fill the idle time between requests in order to enhance LR-DDoS attacks and degrade the target system's performance. Zhang, *et al.* [18] introduced a LR-DDoS attack aimed at border routers, specifically exploiting the transport layer vulnerabilities of BGP. Thus, it is important to address and provide extensible solutions to the detection and mitigation of LR-DDoS attacks in real-world networks.

A LR-DDoS can be broadly characterized by four parameters [19], [20], namely: T_a is the attack period (frequency for sending malicious packets), T_b is the burst width (time duration of the attacking pulse), R_b is the attack burst rate (amount of traffic), and s is the starting time of the attack. This also implies that such an attack can be identified using the source IP and port, destination IP and port, and the protocol used in the attack. Figure 1 shows a LR-DDoS attack with a single source.

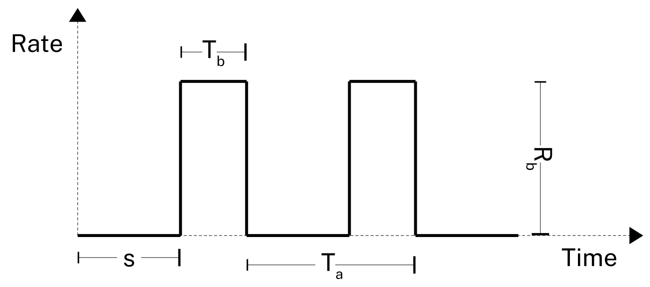


FIGURE 1. LR-DDoS attack model. Burst traffic (T_b) is sent to the target every T_a seconds at a R_b B/s rate, starting at s seconds.

Generally, LR-DDoS are facilitated using multiple sources, say $F_1, F_2, \cdots, F_n$ for each flow. If T_a, T_b, and R_b are equal for each flow F_i, the attack is defined as a group flow (a set of attacks with the same target and characteristics). Combination of group flows (different parameters for each attack) enables more disruptive attacks. Zhang, *et al.* [19] provided a classification for LR-DDoS attacks:

- Attack Frequency Intensification (AFI). The distributed attack has the same parameters but different starting times (s). Thus, the attack has a higher frequency.
- Attack burst Width Intensification (AWI). An attack burst is immediately followed by another attack burst. Therefore, the total attack burst is intensified n times.
- Attack burst Rate Intensification (ARI). If two or more flows start at the same time, the burst rate (R_b) for each flow is aggregated.
- Mixed Intensification (MI). Complex combinations of previous attack type categories.

B. SOFTWARE-DEFINED NETWORKING

SDN is a relatively new networking paradigm, which can help mitigate the limitations of current switching networking by decoupling control and data planes, formerly implemented inside switches and routers, and enabling more flexible and manageable environments [21]. In SDN, the control plane is located in a logically centralized controller, which simplifies policy enforcement and network configuration evolution [22]. There are several benefits of using SDN, such as:

1) network policies are defined using high-level languages in applications instead of low-level, vendor-specific commands;
2) application development is straightforward since the controller provides useful network abstractions, such as global network views, and consequently one achieves more efficient and sophisticated control; and
3) switching devices become multi-purpose devices because they follow flow rules provided by the control layer.

Figure II-B provides an architectural view of SDN.

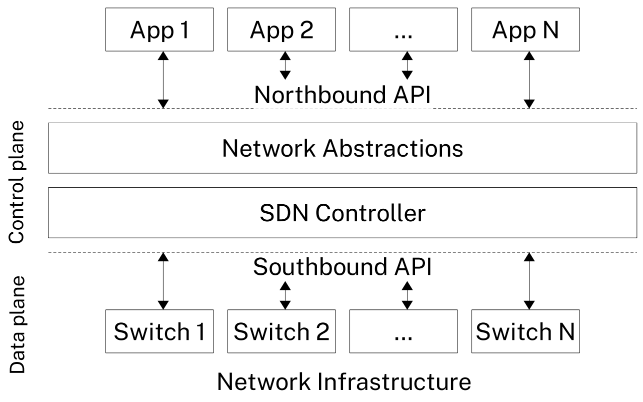


FIGURE 2. SDN architectural view.

The fundamental principles of the SDN architecture are as follows:

1) the decoupling of physical and logical layer in networking devices, allowing each layer to evolve independently, enables innovation, acceleration of new features and services, manageability, among others;
2) devices and users should not be able to differentiate between conventional networks and SDN; and
3) automation and runtime deployment by logically centralizing the control plane and introducing programmable entities.

The above fundamental principles can be achieved in the three-layer architecture presented by the Open Networking Foundation (ONF), which is also described below:

1) The application plane includes a variety of services and applications such as Deep Packet Inspector (DPI), Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), and monitoring. They can inform decision-making in a range of applications such as traffic engineering, quality of service (QoS) differentiation, monitoring, and routing.
2) The control plane is responsible for the management of the underlying forwarding devices by using global network knowledge and information for decision making. It also interacts with the application plane to provide useful information for applications.
3) The data plane includes a variety of forwarding devices such as routers and switches. They forward packets based on flow tables populated by the control plane. It is also responsible for collecting network information and statistics to be later shared with the controller.

In our work, we use a SDN-based architecture, where a solution can be developed through an application that can be installed in the controller for the detection and mitigation of LR-DDoS attacks. This, in turn, makes use of the programmable nature of the network by using new technologies such as machine and deep learning techniques to provide robust mechanisms for detecting and mitigating LR-DDoS attacks.

III. RELATED WORK

Xiang *et al.* [7] proposed a detection mechanism based on Shannon's entropy of information theory as well as a traceback mechanism to detect attackers in a local area network. A generalized entropy metric is used as a mechanism to detect anomalous traffic. The authors assumed that malicious traffic follows a Poisson distribution and normal traffic follows a Gaussian normal distribution, as they argued that the entropy value on Gaussian distributions is higher than that of the Poisson distribution. This entropy value, an indicator of the randomness of a variable, is then used to classify malicious traffic from normal traffic. Also, the probabilities for the computation of the entropy have to be defined beforehand. Experimental results showed that the generalized entropy metric achieves better performance than Shannon's entropy. However, such mechanisms for low-rate attack detection are difficult to implement since several values have to be computed beforehand and, since each network dynamics and topology are different, such values have to be tuned for optimal performance.

Baskar *et al.* [23] argued that entropy is only one of the few feasible parameters to detect low-rate DoS attacks due to its low computing requirements and effectiveness on the study of flow randomness. They then proposed a framework an adaptive IP traceback mechanism for detecting low-rate attacks. In their architecture, an AADS device is placed on each LAN and is responsible for detecting the attacks. When an AADS detects a change in entropy, it communicates with the routers to obtain information about the attacker. The architecture was evaluated through simulation experiments. However, there are several limitations in this approach. For example, AADS network devices need to be placed inside several networks and network traffic models must be known beforehand to properly calculate entropy values.

Kumawat and Meena [24] proposed a framework for the detection and mitigation of low-rate DoS attacks based on information entropy analysis. The framework has three phases, namely: (1) the characterization phase calculates the entropy of each flow and compares them with a set of predefined thresholds; (2) the detection phase classifies the flow as high-rate DoS attack if its entropy is higher than the threshold for that flow, and as low-rate DoS attack if it is lower than the threshold; and (3) the mitigation phase stops the attack near the source. The framework was evaluated using NS-2 (a network simulator), and achieved good results in the mitigation phase. However, there are several drawbacks with this approach. First, there is no clear way to determine the thresholds for each flow as it requires an statistical study of each flow. Moreover, the behavior in entropy of each flow can be determined by the type of generated traffic, the communication protocol used, the communication technology, among others. In addition, normal flows can be classified as low-rate attacks because not all traffic is constant for the reasons stated earlier.

Bhuyan *et al.* [25] provided a comparative evaluation of several information metrics for low-rate DoS attack detection approaches. The authors compared Hartley entropy, Shannon entropy, Renyi's entropy, and Generalized entropy in terms of their ability in detecting low-rate attacks. Two datasets were used to evaluate such metrics, namely: the MIT Lincoln Laboratory for normal traffic and the CAIDA DDoS 2007 for attack traffic datasets. Results showed that for low-rate attacks, increasing the order of generalized entropy provides better results by adjusting the value of order for α in $H_\alpha(x) = \frac{1}{1-\alpha} log_2 \left(\sum_{i=1}^{n} p_i^\alpha\right)$. However, entropy-based solutions require large amounts of data before they can provide good decisions.

Bhuyam *et al.* [26] proposed a mechanism based on correlation coefficients to detect low-rate and high-rate DDoS attacks. Correlation is important in finding linear relationship between two variables. Specifically, partial rank correlation is used to detect low-rate attacks. The detection mechanism is based on the idea that malicious instances (attackers) have correlation coefficients close to one. The mechanism uses a correction based on two thresholds to justify whether packets are malicious or not. Results showed that correlation between two malicious traffic instances is strong. It is, however, not clear if the proposed solution works when just one malicious traffic instance is attacking the network. This limits its potential.

Hoque *et al.* [27] introduced a statistical measure for multivariate data analysis to classify DDoS attack traffic from normal traffic. Three features are selected, namely: the entropy of source IPs, their variation, and the packet size of malicious traffic flows. Normal traffic is used to determine the traffic's normal profile. The captured traffic is then compared against the profiled traffic to classify traffic as either malicious or normal. However, it is not clear how the proposed metric is used to detect low-rate attacks.

Zhang *et al.* [28] proposed the Congestion Participation Rate (CPR) metric to detect and filter low-rate DoS attacks. CPR identifies attacking flows since low-rat attacks actively induce network congestion. CPR is designed to distinguish between normal TCP flows and low-rate attack flows, and it is based on the ratio of incoming packets in congestion to the total incoming packets from certain flow. CPR can be implemented on the front of the Random Early Detection (RED) [29] queue management mechanism of routers. Kieu *et al.* [30] extended the CPR approach based on the argument that CPR sets a fixed threshold for low-rate attack detection, resulting in unfair treatment of new TCP flows that still are to achieve high throughput. The authors also proposed a method to adapt this threshold according to whether the network is under attack or not.

The authors in [31] used the Generalized Total Variation metric for detecting low-rate DDoS attacks. This approach separates legitimate traffic based on sampling values of arriving packets, where wider spacing values indicate low-rate attacks. A value of the generalized total variation metric between two consecutive samples, will trigger an alarm if the value exceeds a threshold δ. Given two systems, if the first system sends more than 500 packets in an interval of more than one second, and the second system sends less than $5,000$ packets in less than 2 seconds, then the system is deemed to be experiencing a multi-scale low-rate DDoS attack. Experiments were carried using the MIT Lincoln Laboratory (attack-free) dataset and the CAIDA DDoS 2007 dataset for attack traffic. Results showed that this approach has a 98.57% attack detection. However, a careful selection of the threshold δ is essential to achieving good performance.

SDN architecture has a programmable feature, in which network operators can develop and deploy applications that run on top of the controller to provide network functionality. This makes the detection and mitigation of several DDoS attacks easier to implement and evaluate on real network deployments. Hong *et al.* [32] introduced the Slow HTTP DDoS Defense Application (SHDA) that runs on top of an SDN controller. On the detection of an incomplete HTTP transaction and the number of open connections on the web server exceeding a certain threshold, the SHDA processes the packets coming from the attacker and determines based on timeouts if the particular traffic is malicious or not. The SHDA installs a new flow rule that blocks the attacker's flow at the switch. This approach provides a basic scheme for SDN-based mitigation techniques for low-rate DoS attacks. However, attackers may dynamically vary the sampling period of HTTP requests, making it difficult to mitigate low-rate attacks based solely on timeouts.

Wu *et al.* [33] stated that low-rate DoS attacks cannot pose a threat in SDN environments since the controller is a powerful machine. However, SDN devices (e.g, OpenFlow switches) have limited capabilities and can become a target for such attacks. In fact, some researchers have already carried out successful attacks against the limited TCAM feature of switches [34]. The TCAM is responsible for storing flow rules dictated by applications running on top of the controller in an SDN. Thus, the authors [33] studied four features, namely: the amount of time a flow rule is present in a switch, the total number of packets matched by a flow rule, the relative dispersion of bytes between normal and attacking flows, and the relative dispersion of packet intervals of arrival. A Factorization Machine algorithm was used to obtain a lineal model of the system based on the input features described earlier. Performance was evaluated using the NSL-KDD, DARPA98, and CAIDA datasets in a simulated environment.

Container-based cloud services are rapidly growing due to its ease of deployment and complete control for customers. In traditional cloud environments, a web application is expected to run as an independent instance on a virtual machine. If a component of the application experiences a DDoS attack, the entire instance is at risk. Container-based cloud environments used alongside with microservice architectures are more effective and agile in resources usage, providing straightforward techniques to scale. As discussed previously, SDN allows the deployment of networks based on software. All implementations of algorithms and protocols are available as software entities instead of firmware entities in closed-source devices. Thus, SDN and container-based environments can be used in conjunction as virtualized networking functions (NFVs), in order to provide more robust and flexible ways of controlling resources for services. Li *et al.* [35] provided a model and mitigation technique to detect and block low-rate attacks on container-based cloud services. They proposed the isolation into two parts of instances, where one part controls requests from a whitelist (legitimate traffic) and the other serves unknown requests (both malicious and benign requests). The mitigation mechanism computes the minimum resources and optimum number of containers for each trusted connection (users with access rights). Then, resources are isolated into containers to avoid resource competition. However, unknown requests are a combination of malicious and normal traffic. The system then guarantees resources for normal traffic in unknown requests while giving minimum resources to malicious traffic. However, instantiating containers according to the amount of traffic required by users affects directly in processing power and memory consumption of the overall system.

Zhang *et al.* [36] provided a low-rate attack detection using Power Spectral Density (PSD) entropy and Support Vector Machines (SVM). They argued that PSD-entropy has low-computation cost and improves detection and efficiency of the system. To classify traffic, two thresholds are calculated by computing the mean of normal traffic and the mean of attacking traffic. If the calculated entropy is lower than the lowest entropy, it is classified as attacking traffic. SVM is

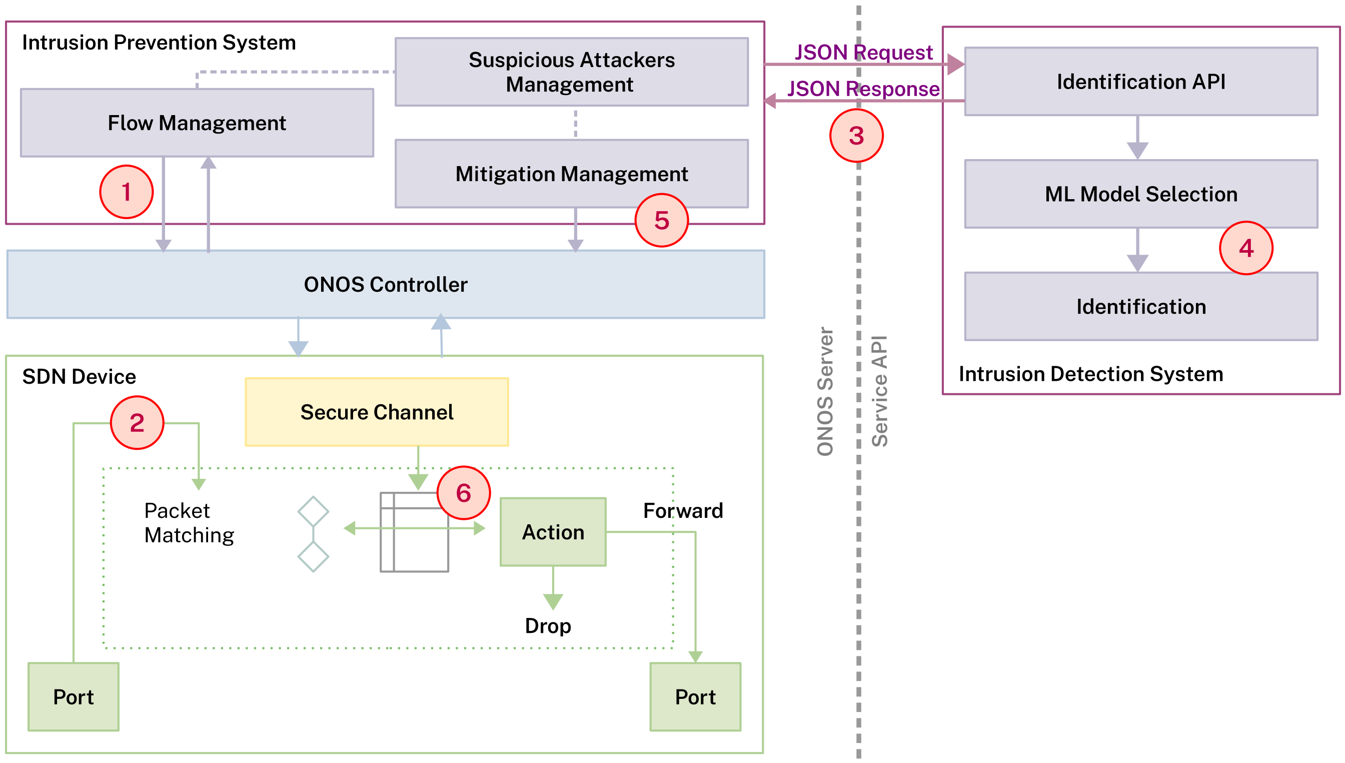


FIGURE 3. **Our proposed framework.**

used to learn traffic patterns and to select appropriate features for the detection algorithm.

Liu *et al.* [8] explained that traffic volume analysis cannot detect current stealthy low-rate DoS attacks. They then proposed a deep convolution neural network (DCNN) to extract available features automatically, and a Q-Network method (a reinforcement learning algorithm) to detect edge low-rate DoS attacks. Results showed that this approach can maintain acceptable network performance in simulated environments.

Meti *et al.* [37] proposed using SVM and Neural Networks (NN) as classifiers for intrusion detection and DDoS attacks in SDN. The approach showed promising results in detecting regular DDoS attacks for NN with 80% accuracy and 100% precision. Similarly, Virupakshar *et al.* [38] evaluated the performance of Decision Trees, K-nearest neighbor (KNN), Naive Bayes, and Deep Neural Network (DNN) in flooding attack detection on an OpenStack-based private cloud. Their findings showed that KNN, Naive Bayes and DNN achieve high accuracy, specially for DNN.

IV. ARCHITECTURAL DESIGN

We will now present our proposed framework designed to mitigate Low-rate DDoS attacks in SDN. Specifically, the framework decouples the detection and mitigation processes from the network application, thus reducing processing requirements from the controller, while being programming language independent and technology-agnostic. Therefore, any programming language and machine / deep learning framework can be used to implement and train different techniques and models to identify different types of Low-rate DDOS attacks. Moreover, our approach allows machine / deep learning techniques to fully utilize GPU for faster training and classification, since such applications are standalone processes.

As shown in Figure 3, the framework comprises two independent systems, namely: an *Intrusion Prevention System (IPS)* and an *Intrusion Detection System (IDS)*.

The IPS consists of three sub-modules:

1) the *Flow Management* module is responsible for detecting HTTP flows for further processing;
2) the *Suspicious Attackers Management* manages a blacklist of potential attackers; and
3) the *Mitigation Management* module generates flow rules for malicious flow mitigation.

The IDS also consists of three sub-modules, which are described below:

1) the *Identification API* provides an interface for the interaction between the IPS and IDS systems;
2) the *ML Model Selection* consists of a set of trained ML models used for flow identification, and
3) the *Identification* performs malicious flow classification.

We remark that the IPS is executed on top of the ONOS controller and the IDS is executed on a separate host (for demonstration, we used Windows 10 in our experimental setup). In practice, the IDS can be deployed at any remote host with any operating system and system libraries. However, for optimal performance its deployment should be close to the IPS (that runs on the top of the controller) to avoid latency due to bandwidth limitations.

Figure 3 shows the steps of a typical scenario in which a potential threat is mitigated:

1) The *IPS* is a network application running on top of the ONOS Controller. In this stage, the *Flow Management* module is responsible for installing flow rules inside the SDN Device for HTTP header detection. The SDN Device requires a flow rule installed in order to detect HTTP flows and to forward them to the controller and subsequently to the Flow Management module.
2) A flow incoming from a physical Port at the SDN Device is matched against the flow table entry installed by the Flow Management module, seeking for HTTP headers. If a match exists, the SDN Device forwards through a Secure Channel the flow to the SDN Controller and later to the Flow Management module.
3) The Flow Management module creates a JSON object with the headers from the flow to be later forwarded to the *Identification API* for further inspection indicating which ML model should be use.
4) The Identification API selects the proper model from the *ML Model Selection* module and forwards the flow headers to the *Identification* module for classification as an attack or normal flow. The IDS sends a proper JSON response to the IPS on the controller.
5) The *Suspicious Attackers Management* module keeps a blacklist of the attackers previously identified by the IDS. Further explanation of this module is provided in Section IV-B. The *Mitigation Management* is constantly looking for high values of probabilities(100%) in the Suspicious Attackers list, in order to create the proper flow rules for attack mitigation.
6) Finally, the previously created rules are installed inside the SDN Device through the ONOS Controller and the Secure Channel to mitigate a host (ab)used to carry out the attack.

The Identification API allows researchers and operators to implement a wide range of machine / deep learning algorithms since different techniques are best suited for different scenarios. The logical separation of the IDS from the controller allows one to use different hardware architectures for faster machine / deep learning model training and processing. Moreover, the framework provides a reference model for future machine / deep learning-based techniques not only for security, but also for other network operations such as optimal path finding in large-scale networks.

Currently, the proposed framework employs an anomaly-based IDS, where we use a defined model of normal network behavior (trained machine / deep learning models) in order to detect deviations from such a model [39]. We can, however, employ a hybrid IDS based on both anomaly and signature approaches in order to provide a more robust identification system. In such a scenario, the header information received by the Identification API is extracted as signatures and is then compared against patterns or rules and complex regular expressions (RegEx). If no matches are found, the header information is then processed by the anomaly-based IDS that involves machine / deep learning techniques. The information flows as a daisy chain process passing first through the signature-based IDS approach and, if does not match, through the anomaly-based IDS. Such hybrid IDS remains a topic of ongoing research interest [40]–[43] and can further be enhanced and tested by implementing modular architectures such as our proposed framework.

A. IDS AND IDENTIFICATION API

The *Identification API* provides an interface for flow processing using trained machine / deep learning models. The interface defines a *Classify* object that performs the identification of a `Flow` attribute using a `Classifier`. The classifier is a string that represents the different trained ML models available in the IDS. The flow complex object gathers flow statistics from SDN Devices for classification as an attack or a legitimate flow. The attributes on this flow complex object are based on **flowtbag**.[1] Algorithm 1 shows the process performed for the classification of flows inside the IDS module.

Algorithm 1: IDS Classification Process

```
input  : JSON request with traffic flow parameters
output: JSON response with flow classification
1  Receives a set of flow parameter inputs;
2  Selects the ML model specified in the JSON request;
3  Classifies the input parameters with the selected model;
4  if flow is classified as anomalous then
5      Detects the type of attack;
6      Creates a JSON response with the attack type as
         data;
7      Sends the JSON response to the IPS;
8  else
9      Creates a JSON response with attack type as 0;
10     Sends the JSON response to the IPS;
11 end
```

As stated before, the framework allows the implementation of different ML models. We were able to implement the following algorithms to evaluate their accuracy in detecting LR-DDoS attacks:

- *J48* provided by Weka, an open source, machine learning library written in Java. J48 implements C4.5 decision trees [44] for classification;
- *Random Tree* provided by Weka. It considers K randomly chosen attributes at each node for classification;

[1] `https://github.com/DanielArndt/flowtbag/wiki/features`

- *REPTree* provided by Weka. It builds a decision/regression tree using information gain/variance with reduced-error pruning;
- *Random Forests*. Tree predictors for classification [45].
- *Multi-Layer Perceptron (MLP)* provided by TensorFlow 2.0 and Scikit-Learn 0.23. MLP has been proven to provide good accuracy in detecting DDoS attacks [46]; and
- *Support Vector Machines (SVM)* provided by Scikit-Learn 0.23. SVM has proven to be an effective classification technique for DoS attacks in SDN environments [47].

From our literature review, we observe that MLP is widely used in several works [48]–[54] as a promising detection technique for traditional DDoS attack detection. Also, as far as we know MLP algorithm is ideal for tabular datasets or presented data like those coming from our dataset. Studies like Kim and Gofman [55] show that MLP yields better performance compared to other Deep Neural Networks. Therefore, we chose MLP for the evaluation and validation of the proposed framework considering some other deep learning algorithms as future work.

The above algorithms implemented in our architecture were adapted to effectively identify malicious flows in LR-DDoS attacks. As mentioned in Section I, the included attacks in the Dataset which are later identified are listed below:

- *DDoSSim* a layer 7 DDoS simulator
- *GoldenEye*
- *H.U.L.K* the HTTP Unbreakable Load King
- *R.U.D.Y*, R U Dead yet DoS tool
- *SlowBody2*
- *Slow Headers*
- *Slowloris*
- *Slowread*

We used the CIC DoS Dataset (2017) [56] to train the algorithms. The *Identification API* returns a JSON object with 0 if the flow is classified as legitimate, and numbers $[1-8]$ for each of the attack types listed earlier.

B. IPS AND MITIGATION STRATEGY

As we have already explained, an IDS API was initially developed to request flows regularly and analyze them by invoking a previously AI trained model. The flows are sent by the *Flow Management* module, which also received the reply of the IDS API. In this first version of the IPS of our architecture, we decided to implement an efficient and not overwhelming method that makes it easier for scalability. Algorithm 2 shows the process of extracting features from incoming flows.

The IDS API will return a 0 if it is a normal flow, otherwise it will return a number that correspond to the one specific LR-DDoS attack that was detected. Once the IDS returns an attack match for a tested flow indicating that the source of this flow could potentially be an attacker, the *Flow Management*

Algorithm 2: IPS Flow Management

```
input  : Traffic flows
output: JSON request with traffic flow parameters
1 Extracts flow and header features from incoming traffic;
2 Creates a JSON request with flow and header features;
3 Sends the JSON request to the Identification API;
```

module forwards the information of the potential attacker to *Suspicious Attacker Management* module where the source IP address is blacklisted in a blacklist (flow drop probability table) of potential attackers and would be added with a flow drop probability in the control plane of 10%. If the IP address is already in the blacklist when it is received by the *Suspicious Attacker Management* module then the source IP address will increase its flow drop probability in 5% and so on. It means that the IPS will increment the flow drop probability in 5% for every flow that match with an attack by the IDS. Algorithm 3 illustrates the process of mitigating an attacker.

Algorithm 3: IPS Mitigation Management

```
input  : JSON response with flow classification
output: Flow rule
1  Extracts classification data from JSON data;
2  if flow classification is anomalous then
3      The source IP address is appended to a blacklist;
4  end
5  if source IP address already on list then
6      if source IP address probability $\geq$ 100 then
7          The controller issues a drop rule with the
             accumulated probability of dropping the flow;
8      else
9          Increases in 5% the probability on a Flow drop
             table;
10     end
11 end
```

Before an attacker's IP reaches the threshold of 30% in the flow drop probability table, the flows of that IP will be forwarded properly preventing the blocking of legitimate hosts that could have been identified mistakenly as an attacker. After passing the 30% threshold, the flows of the possible attacker will be forwarded throughout a dropping function that will drop approximately a number of flows according to the percentage that this IP has in the drop probability table. It implies that this function for example will drop approximately 6 of every 10 flows for an IP which has a flow drop probability of 60%.

If a 100% flow drop probability is reached by a host, the *Suspicious Attacker Management* module will forward its information to the *Mitigation Management* module where a blocking port flow rule will be immediately created and sent to the proper switch, dropping all flows matching that

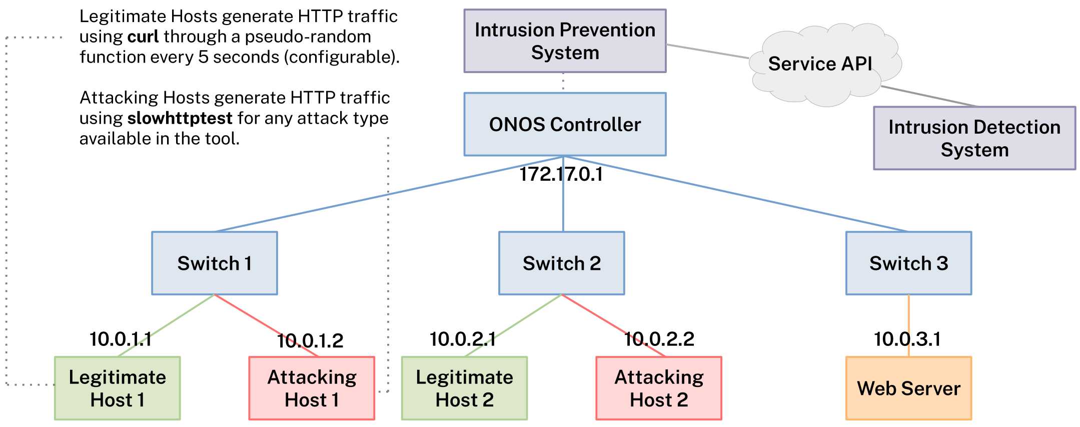

Legitimate Hosts generate HTTP traffic using **curl** through a pseudo-random function every 5 seconds (configurable).
Attacking Hosts generate HTTP traffic using **slowhttptest** for any attack type available in the tool.
Intrusion Prevention System
Service API
Intrusion Detection System
ONOS Controller
172.17.0.1
Switch 1
Switch 2
Switch 3
10.0.1.1
10.0.1.2
10.0.2.1
10.0.2.2
10.0.3.1
Legitimate Host 1
Attacking Host 1
Legitimate Host 2
Attacking Host 2
Web Server

FIGURE 4. **Experimental virtualized network topology.**

IP address, and its Destination TCP/UDP port. After seven days with manual intervention by the administrator, the rule will expire and the host's flow drop probability will be reset to 0%, and finally, the traffic of this host will be analyzed again.

Due to its modular design, our framework allows to locate the IDS outside the controller in a separate hardware component. This design decision follows the idea that the controller should not perform complex tasks such as flow classification. Thus, the controller only needs to focus on processing incoming flows and makes decisions about those flows according to flow rules. For this reason, in our testbed the controller stays stable without any significant changes while running simultaneously the IPS and IDS compared to running only the IDS. Also, it is important to mention that the IPS does not demand a lot of resources since the algorithm is very efficient as shown in Algorithm 3.

V. EXPERIMENTAL SETUP

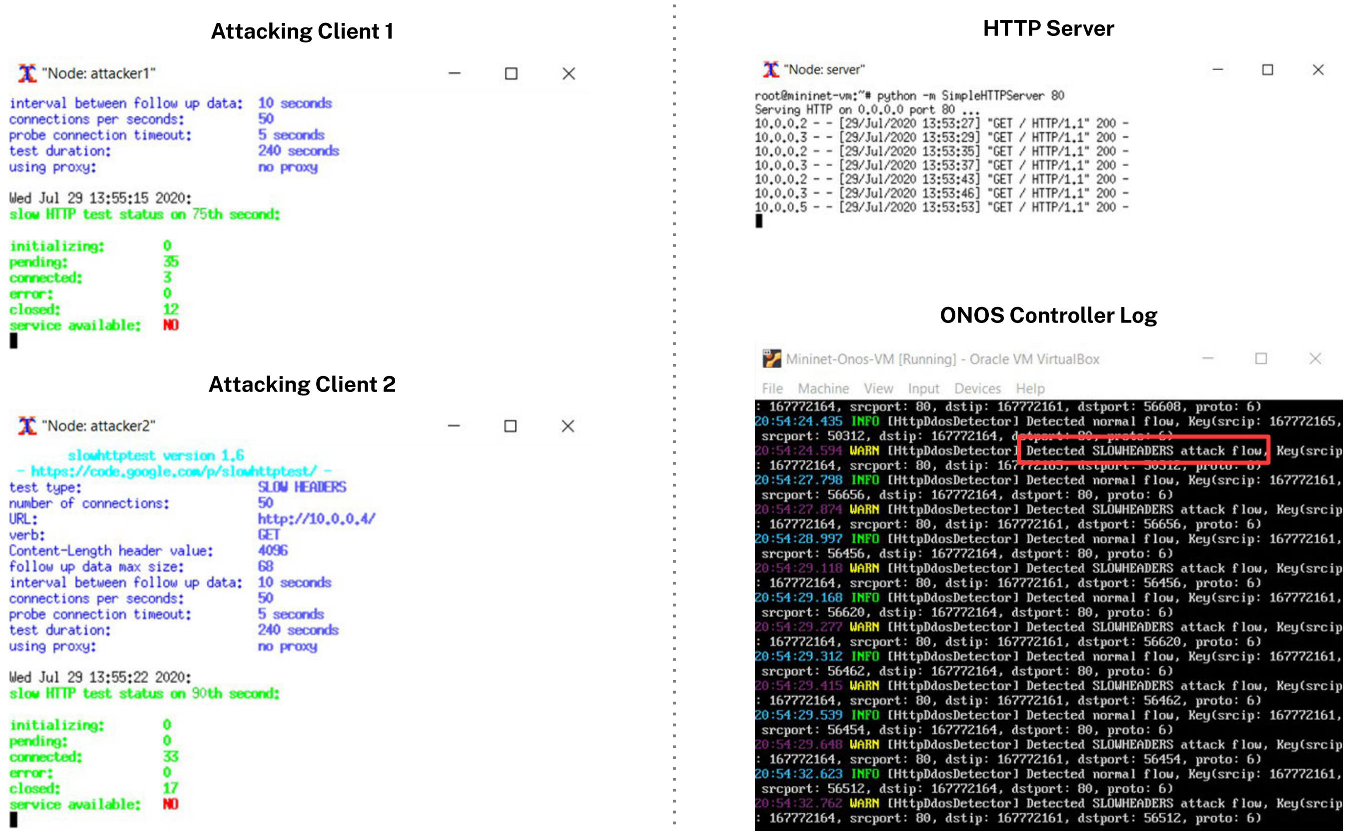
In order to evaluate the viability and the functionality of our framework, we developed a diverse portfolio of machine and deep learning classifiers. In this section we describe the experimental setup and technologies used to evaluate the performance of different ML techniques and the viability of the framework.

A. VIRTUAL ENVIRONMENT SETUP

The SDN environment is emulated using *Mininet*, a helpful tool that enables the creation of virtual network topologies. Virtualized hosts were configured as legitimate hosts and Web servers, where LR-DDoS attacks were launched from other virtualized hosts. The SDN Devices (network switches) are controlled by an ONOS Controller. Figure 5 shows a screenshot of the initial implementation of the architecture on the virtualized ONOS environment. We chose to use VirtualBox for virtualization because the ONOS Project conveniently provides a dockerized ONOS environment that is straightforward to use and ready to start the development and deployment of projects such as the one described in this manuscript. In this initial testbed, we can see how the IDS API is identifying the *Slow Headers attack* which is performed by two attacking clients while two other legitimate clients are sending normal traffic flows in the same network. The main goal of this first approach was to test the IDS and the basic functionality. The testbed implementation was further improved, enabling the use of different administrative domains (different networks) as depicted in Figure 4. This figure shows a based topology used to implement several tests including more hosts and devices. Moreover, implementing the proposed architecture using virtualized environments and ONOS as the controller, allows for straightforward deployment into real network architectures, since ONOS has been used in production environments.

As shown in Figure 3, the IPS runs on top of the ONOS Controller whilst the IDS runs on a separate host inside the network. The controller and the IDS communicate through the Identification API as explained in Section IV. The *SlowHTTPTest* tool is used to launch LR-DDoS attacks from the attackers to the virtualized Web server. The IPS then detects such flows and passes the information to the IDS for further processing. Figure 4 depicts the network topology employed for experimental purposes.

From Figure 4 we observe that the hosts on each of the switches belong to different networks. The *Web Server* on *Switch 3* is the target of *Attacking Host 1* and *Attacking Host 2* belonging to *Switch 1* and *Switch 2* respectively. We assume that the *attacking hosts* are compromised and they could even be part of a botnet [57], where each attacking host can generate malicious traffic using the *SlowHTTPTest* tool. *Legitimate Host 1* and *Legitimate Host 2* generate normal flows with a pseudo random function using *curl* targeting the Web Server. All switches in the topology are being controlled by the *ONOS Controller* (management address). It is worth noting that, we opted to target Web servers for our experimental evaluation since they are one of the most used and vulnerable services on the Internet. Moreover, some of the LR-DDoS attack detection mechanisms target HTTP specifically and web servers normally have less resources than the controller, so if the architecture is able to protect the web servers it will be able to protect the controller as well.



```
interval between follow up data:  10 seconds
connections per seconds:          50
probe connection timeout:         5 seconds
test duration:                    240 seconds
using proxy:                      no proxy

Wed Jul 29 13:55:15 2020:
slow HTTP test status on 75th second:

initializing:      0
pending:           35
connected:         3
error:             0
closed:            12
service available: NO
```

```
        slowhttptest version 1.6
 - https://code.google.com/p/slowhttptest/ -
test type:                        SLOW HEADERS
number of connections:            50
URL:                              http://10.0.0.4/
verb:                             GET
Content-Length header value:      4096
follow up data max size:          68
interval between follow up data:  10 seconds
connections per seconds:          50
probe connection timeout:         5 seconds
test duration:                    240 seconds
using proxy:                      no proxy

Wed Jul 29 13:55:22 2020:
slow HTTP test status on 90th second:

initializing:      0
pending:           0
connected:         33
error:             0
closed:            17
service available: NO
```

```
root@mininet-vm:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
10.0.0.2 - - [29/Jul/2020 13:53:27] "GET / HTTP/1.1" 200 -
10.0.0.3 - - [29/Jul/2020 13:53:29] "GET / HTTP/1.1" 200 -
10.0.0.2 - - [29/Jul/2020 13:53:35] "GET / HTTP/1.1" 200 -
10.0.0.3 - - [29/Jul/2020 13:53:37] "GET / HTTP/1.1" 200 -
10.0.0.2 - - [29/Jul/2020 13:53:43] "GET / HTTP/1.1" 200 -
10.0.0.3 - - [29/Jul/2020 13:53:46] "GET / HTTP/1.1" 200 -
10.0.0.5 - - [29/Jul/2020 13:53:53] "GET / HTTP/1.1" 200 -
```

```
: 167772164, srcport: 80, dstip: 167772161, dstport: 56608, proto: 6)
20:54:24.435 INFO [HttpDdosDetector] Detected normal flow, Key(srcip: 167772165,
 srcport: 50312, dstip: 167772164, dstport: 80, proto: 6)
20:54:24.594 WARN [HttpDdosDetector] Detected SLOWHEADERS attack flow, Key(srcip
: 167772164, srcport: 80, dstip: 167772165, dstport: 50312, proto: 6)
20:54:27.798 INFO [HttpDdosDetector] Detected normal flow, Key(srcip: 167772161,
 srcport: 56656, dstip: 167772164, dstport: 80, proto: 6)
20:54:27.874 WARN [HttpDdosDetector] Detected SLOWHEADERS attack flow, Key(srcip
: 167772164, srcport: 80, dstip: 167772161, dstport: 56656, proto: 6)
20:54:28.997 INFO [HttpDdosDetector] Detected normal flow, Key(srcip: 167772161,
 srcport: 56456, dstip: 167772164, dstport: 80, proto: 6)
20:54:29.118 WARN [HttpDdosDetector] Detected SLOWHEADERS attack flow, Key(srcip
: 167772164, srcport: 80, dstip: 167772161, dstport: 56456, proto: 6)
20:54:29.168 INFO [HttpDdosDetector] Detected normal flow, Key(srcip: 167772161,
 srcport: 56620, dstip: 167772164, dstport: 80, proto: 6)
20:54:29.277 WARN [HttpDdosDetector] Detected SLOWHEADERS attack flow, Key(srcip
: 167772164, srcport: 80, dstip: 167772161, dstport: 56620, proto: 6)
20:54:29.312 INFO [HttpDdosDetector] Detected normal flow, Key(srcip: 167772161,
 srcport: 56462, dstip: 167772164, dstport: 80, proto: 6)
20:54:29.415 WARN [HttpDdosDetector] Detected SLOWHEADERS attack flow, Key(srcip
: 167772164, srcport: 80, dstip: 167772161, dstport: 56462, proto: 6)
20:54:29.539 INFO [HttpDdosDetector] Detected normal flow, Key(srcip: 167772161,
 srcport: 56454, dstip: 167772164, dstport: 80, proto: 6)
20:54:29.648 WARN [HttpDdosDetector] Detected SLOWHEADERS attack flow, Key(srcip
: 167772164, srcport: 80, dstip: 167772161, dstport: 56454, proto: 6)
20:54:32.623 INFO [HttpDdosDetector] Detected normal flow, Key(srcip: 167772161,
 srcport: 56512, dstip: 167772164, dstport: 80, proto: 6)
20:54:32.762 WARN [HttpDdosDetector] Detected SLOWHEADERS attack flow, Key(srcip
: 167772164, srcport: 80, dstip: 167772161, dstport: 56512, proto: 6)
```

FIGURE 5. **Initial experimental implementation. The controller is detecting the SlowHeaders attack (highlighted in red) from attacking clients.**

Once the topology is setup, the *Legitimate Host 1* and *Legitimate Host 2* start to send normal traffic. At the same time, the *Attacking Host 1* starts sending LR-DDoS attacks while the *Attacking Host 2* starts by sending legitimate traffic and after a while it becomes an attacker and starts attacking as well with the *Attacking Host 1* to create a LR-DDoS attack. During the attack we used different kind of attacks provided by *SlowHTTPTest* utiliy (such as *Slowloris, Slow headers, R.U.D.Y.*, etc.). Reactive Forwarding is enabled on the ONOS Controller for packet forwarding between network switches. Furthermore we tested some other testbed settings by increasing the number of hosts and devices in the architecture, where the framework shows an stable and acceptable performance.

B. DATASET

Traditional DoS/DDoS attacks are characterized by a high volume of application-layer requests. On the other hand, low-volume or LR-DDoS attacks employ minimal traffic transmitted strategically. While there are a small number of publicly available datasets, namely the NSL-KDD, CAIDA, and CIC DoS 2019 datasets, only the 2017 CIC DoS dataset [56] captures LR-DDoS attacks. The NSL-KDD dataset focuses mainly on remote to local, user to root and general DDoS attacks, while CIC DoS 2019 dataset includes reflection and exploitation DoS attacks.

TABLE 1. **Dataset number and percentage of flows by type.**

Type of traffic	Number of flows	Percentage
Legitimate	113, 438	85.68213%
DDoSSim	2, 185	1.650375%
GoldenEye	661	0.499268%
H.U.L.K.	2, 088	1.577111%
R.U.D.Y.	2, 070	1.563515%
Slowbody2	4, 391	3.316616%
Slowheaders	3, 185	2.405699%
Slowloris	2, 912	2.199496%
Slowread	1, 464	1.10579%

We also need data to train the models, and therefore a compilation of data flows with both normal traffic and LR-DDoS attacks are used. Specifically, we used the CIC DoS Dataset (2017) [56], which contains regular traffic (labeled as normal) and eight low rate attack variations. The features *source IP*, *destination IP*, *ports*, and *protocol* are removed from the original dataset since they do not add relevant information about the LR-DDoS attacks. Moreover, the original dataset consists of packets over a conventional network, and *flowtbag* was used to convert the packet data as flow data, to adapt the dataset to an SDN environment. Flowtbag takes as input a set of packets with 44 features, and outputs flows with the same 44 features. Table 1 shows the distribution of the different types of traffic available in the dataset.

Readers interested in the complete list and description are referred to [58].

C. HYPERPARAMETER OPTIMIZATION

Two different approaches for hyperparameter optimization were used depending on the ML techniques. The random search technique gathers samples from the search space and evaluates sets from a specified value-range with a uniform probability distribution. It was also used to find the optimal values for parameters as activation functions, maximum number of iterations, and the size of hidden layers. The gradient search technique uses an exhaustive search for every possible value of every given combinations. The combination of random and grid searches are used to train the parameters of each classifier, the general training parameters including all the algorithms are shown in Table 2.

TABLE 2. Optimal training parameters.

Algorithm	Training Parameters
J48	Two instances per leaf, three folds for reduced error pruning, one as seed for random data shuffling.
Random Tree	Twelve random attributes to investigate, 27 minimum instances per leaf, 0.0001 minimum class variance proportion, one as seed for random number generator, 23 maximum depth of the tree, five folds for backfitting, and randomly break ties when several attributes look equally good.
REP Tree	28 as maximum tree depth.
Random Forest	Infinite depth, 50 trees maximum, seven random fields per tree.
SVM	RBF kernel classifier and $C = 7.303885828086103$, $\gamma = 0.08590489412933254$.
MLP	30 epochs, five layers (one input, three hidden, one output).

Further optimizations on SVM include the training of different kernels, including linear, polynomial, Radial Basis Function (RBF), and Sigmoid kernel using a subset of the dataset. RBF kernel classifier is selected since it shows better accuracy score. Table 3 shows results for SVM accuracy using different kernels. For MLP we found the best performance using 30 epochs, five layers (one input, three hidden, one output)

TABLE 3. SVM accuracy score.

Kernel	Score
Linear	85%
Polynomial	86%
Radial Basis Function	87%
Sigmoid kernel	83%

It is worth noting that we also implemented algorithms such as a Simple 3-layer Neural Network and AdaBoostM1, which did not achieve acceptable performance. Thus, such algorithms are not included in evaluation performance testing.

VI. FINDINGS

We will now describe the findings from the evaluations based on the setup described in Section V. Each algorithm is evaluated in terms of accuracy, false alarm rate, precision, recall, and F1-measure. An ideal IDS should achieve high accuracy, precision, recall, and F1-measure with low false alarm rate.

Accuracy is computed as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP} \tag{1}$$

In the above equation, *true positive* (TP) denotes the correctly classified malicious flow, *true negative* denotes the correctly classified normal flow, *false negative* (FN) is the incorrectly classified normal flow, and *false positive* (FP) is the incorrectly classified attacking flow.

False alarm rate is calculated as follows:

$$\text{False Alarm Rate} = \frac{FP}{TN + FP} \tag{2}$$

Precision is computed as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

Recall is calculated as follows:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

Finally, the F1-measure is computed as follows:

$$\text{F1-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

The findings of the evaluation are depicted in Table 4 and Figure 6. We can observe that in terms of LR-DDoS attack detection, Random Forest achieves an accuracy rate of 94.41% and false alarm rate of 3.56%, SVM with 93.1% accuracy and 1.6% false alarm rate, and MLP with 95.01% accuracy and 0.52% false alarm rate.

In the evaluation, we also attempted the different attacks supported by the SlowHTTPTest tool (Slowloris, SlowHeaders, R.U.D.Y., etc.), and the IPS successfully blocked attacks previously identified by the IDS. Thus, the IPS effectively blocks attacks that previously have a 100% in the drop probability table as discussed in Section IV-B.

One can also observe that the IDS can effectively detect whether a flow is anomalous and support attack classification for generic (i.e., attacks outside the provided categories), SlowBody, SlowRead, DDoSSim, SlowHeaders, GoldenEye, R.U.D.Y., H.U.L.K. and Slowloris.

In conventional DoS and DDoS attacks, ML algorithms generally seek for patterns to classify flows as malicious or legitimate. Due to the nature of LR-DDoS attacks, DL algorithms such as MLP appear to perform better since their hidden layers allow for features (e.g., connection duration and memory footprint) to be used to inform classification.

Moreover, our findings described in this section show that the proposed framework is robust and flexible in LR-DDoS attack detection and mitigation. The IDS being the most

TABLE 4. Performance metrics by algorithm.

Algorithm	Accuracy	False Alarm Rate	Precision	Recall	F1-measure
J48	0.9068	0.0397	0.6522	0.528	0.5836
Random Tree	0.9176	0.03	0.7283	0.5565	0.6309
REP Tree	0.9037	0.0398	0.6417	0.5044	0.5648
Random Forest	0.9441	0.0359	0.7833	0.8186	0.8005
SVM	0.931	0.016	0.92	0.93	0.93
MLP	0.9501	0.0052	0.9546	0.9451	0.9498

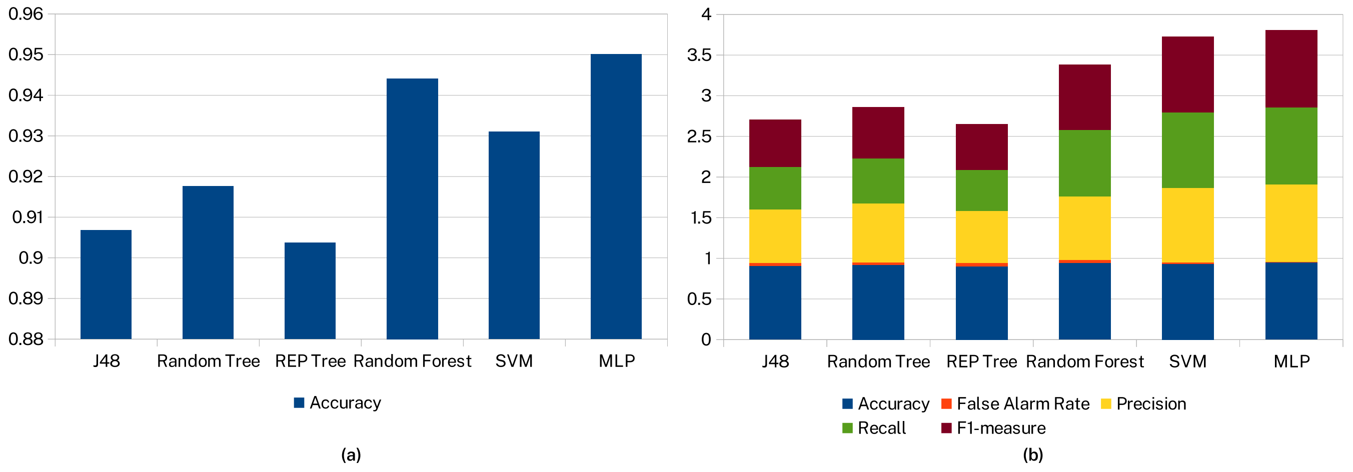


FIGURE 6. (a) Accuracy estimation results by algorithm. (b) Evaluation metric results.

processing-intensive module is located outside the SDN Controller. In other words, the IDS is isolated from the controller, and therefore performing at rates close to an inline, hardware-based IDS since such module can be executed as a virtualized component with NFV or as a hardware-based component.

VII. CONCLUSION AND FUTURE WORK

LR-DDoS attacks are likely to remain a threat to our systems, particularly those that are centralized (e.g., cloud computing platforms). In this paper, we designed and implemented a modular and flexible security architecture to detect and mitigate LR-DDoS attacks in SDN environments. The modularity of the design allows one to easily replace any module without affecting the other modules of the architecture. The IDS module in our architecture is designed to detect flows using different previously trained ML models, which can be developed using different programming languages and frameworks. Findings from the evaluations of the six different ML algorithms using the CIC DoS dataset reported an accuracy rate of 95%. We also deployed our architecture using a real virtualized environment using Mininet virtual machine over VirtualBox and the ONOS controller. We also used the (complex) ONOS controller since this controller is widely used in production environments and specially in datacenters, in our evaluations so that the results can easily migrate to a real-world production environment. In our deployment, we used two different topologies and demonstrated that all attacks previously identified by the IDS were successfully mitigated.

In the future, we intend to extend this work to include newer ML and deep learning techniques, with the aim of improving the performance for example against other attacks. In order to provide a more robust evaluation of the framework, we plan to include more deep learning algorithms as they yield promising results on LR-DDoS attack detection. For example to improve the mitigation strategy, can we use statistical filters such as Exponentially Weighted Moving Average (EWMA) [59] and Kalman filters [60] to facilitate decision making in terms of flow rule installation. The goal of such techniques is to avoid blocking legitimate users when the false positive rate increases. In terms of scalability, we also plan to include a selective testing mechanism of flows from the IPS to the IDS. Such an approach is likely to be interoperable with big network topologies and real-world production networks in datacenters.

ACKNOWLEDGMENT

The authors would like to thank José Abraham Torres, Abel Daniel Yañez, Bernardo Laing, and Leví Carbellido for their assistance in the testing and deployment of the architecture. They also thank the reviewers for their invaluable feedback.

REFERENCES

[1] A. Kuzmanovic and E. W. Knightly, "Low-rate TCP-targeted denial of service attacks: The shrew vs. The mice and elephants," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, New York, NY, USA, 2003, p. 75.

[2] A. Shevtekar, K. Anantharam, and N. Ansari, "Low rate TCP denial-of-service attack detection at edge routers," *IEEE Commun. Lett.*, vol. 9, no. 4, pp. 363–365, Apr. 2005.

[3] X. Luo and R. K. C. Chang, "On a new class of pulsing denial-of-service attacks and the defense," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2005, pp. 61–79.

[4] E. Adi, Z. Baig, C. P. Lam, and P. Hingston, "Low-rate Denial-of-Service attacks against HTTP/2 services," in *Proc. 5th Int. Conf. IT Converg. Secur. (ICITCS)*, Aug. 2015, pp. 1–5.

[5] N. Agrawal and S. Tapaswi, "Defense mechanisms against DDoS attacks in a cloud computing environment: State-of-the-Art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3769–3795, Oct. 2019.

[6] O. Osanaiye, K.-K.-R. Choo, and M. Dlodlo, "Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework," *J. Netw. Comput. Appl.*, vol. 67, pp. 147–165, May 2016.

[7] Y. Xiang, K. Li, and W. Zhou, "Low-rate DDoS attacks detection and traceback by using new information metrics," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 2, pp. 426–437, Jun. 2011.

[8] Z. Liu, X. Yin, and Y. Hu, "CPSS LR-DDoS detection and defense in edge computing utilizing DCNN Q-Learning," *IEEE Access*, vol. 8, pp. 42120–42130, 2020.

[9] M. S. Bonfim, K. L. Dias, and S. F. L. Fernandes, "Integrated NFV/SDN architectures: A systematic literature review," *ACM Comput. Surveys*, vol. 51, no. 6, pp. 1–39, Feb. 2019.

[10] J. Cao, Q. Li, R. Xie, K. Sun, G. Gu, M. Xu, and Y. Yang, "The crosspath attack: Disrupting the $*SDN*$ control channel via shared links," in *Proc. 28th Secur. Symp.*, 2019, pp. 19–36.

[11] D. Tang, R. Dai, L. Tang, and X. Li, "Low-rate DoS attack detection based on two-step cluster analysis and UTR analysis," *Hum.-centric Comput. Inf. Sci.*, vol. 10, no. 1, Dec. 2020.

[12] T. Apostolovic, N. Stankovic, K. Milenkovic, and Z. Stanisavljevic, "DDoSSim–System for visual representation of the selected distributed denial of service attacks," in *Proc. Zooming Innov. Consum. Technol. Conf. (ZINC)*, May 2018, pp. 118–122.

[13] J. Seidl. (2013). *Goldeneye Layer 7 (Keepalive+Nocache) Dos Test Tool*. [Online]. Available: https://github.com/jseidl/GoldenEye

[14] Dominus. (2018). *Hulk Ddos Attack Script Created Using Python Libs*. [Online]. Available: https://github.com/Mr4FX/Hulk-ddos-attack

[15] A. Kuzmanovic and E. W. Knightly, "Low-rate TCP-targeted denial of service attacks and counter strategies," *IEEE/ACM Trans. Netw.*, vol. 14, no. 4, pp. 683–696, Aug. 2006.

[16] W. Zhijun, L. Wenjing, L. Liang, and Y. Meng, "Low-rate DoS attacks, detection, defense, and challenges: A survey," *IEEE Access*, vol. 8, pp. 43920–43943, 2020.

[17] H. Li, J. Zhu, Q. Wang, T. Zhou, H. Qiu, and H. Li, "LAAEM: A method to enhance LDoS attack," *IEEE Commun. Lett.*, vol. 20, no. 4, pp. 708–711, Apr. 2016.

[18] Y. Zhang, Z. Morley Mao, and J. Wang, "Low-rate TCP-targeted dos attack disrupts Internet routing," in *Proc. 14th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2007, pp. 1–15.

[19] C. Zhang, Z. Cai, W. Chen, X. Luo, and J. Yin, "Flow level detection and filtering of low-rate DDoS," *Comput. Netw.*, vol. 56, no. 15, pp. 3417–3431, Oct. 2012.

[20] Y. Tarasov, E. Pakulova, and O. Basov, "Modeling of low-rate DDoS-attacks," in *Proc. 12th Int. Conf. Secur. Inf. Netw.*, 2019, pp. 1–7.

[21] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[22] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, Feb. 2013.

[23] M. Baskar, T. Gnanasekaran, and S. Saravanan, "Adaptive IP traceback mechanism for detecting low rate DDoS attacks," in *Proc. IEEE Int. Conf. Emerg. Trends Comput., Commun. Nanotechnol. (ICECCN)*, Mar. 2013, pp. 373–377.

[24] H. Kumawat and G. Meena, "Characterization, detection and mitigation of low-rate DoS attack," in *Proc. Int. Conf. Inf. Commun. Technol. Competitive Strategies*, 2014, pp. 1–5.

[25] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Information metrics for low-rate DDoS attack detection: A comparative evaluation," in *Proc. 7th Int. Conf. Contemp. Comput. (IC3)*, Aug. 2014, pp. 80–84.

[26] M. H. Bhuyan, A. Kalwar, A. Goswami, D. K. Bhattacharyya, and J. K. Kalita, "Low-rate and high-rate distributed DoS attack detection using partial rank correlation," in *Proc. 5th Int. Conf. Commun. Syst. Netw. Technol.*, Apr. 2015, pp. 706–710.

[27] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "A novel measure for low-rate and high-rate DDoS attack detection using multivariate data analysis," in *Proc. 8th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2016, pp. 1–2.

[28] C. Zhang, Z. Cai, W. Chen, X. Luo, and J. Yin, "Flow level detection and filtering of low-rate ddos," *Comput. Netw.*, vol. 56, p. 3417–3431, Oct. 2012.

[29] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, 1993.

[30] M. V. Kieu, D. T. Nguyen, and T. T. Nguyen, "Using CPR metric to detect and filter low-rate DDoS flows," in *Proc. 8th Int. Symp. Inf. Commun. Technol.*, 2017, p. 325.

[31] M. H. Bhuyan and E. Elmroth, "Multi-scale low-rate DDoS attack detection using the generalized total variation metric," in *Proc. 17th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2018, pp. 1040–1047.

[32] K. Hong, Y. Kim, H. Choi, and J. Park, "SDN-assisted slow HTTP DDoS attack defense method," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 688–691, Apr. 2018.

[33] W. Zhijun, X. Qing, W. Jingjie, Y. Meng, and L. Liang, "Low-rate DDoS attack detection based on factorization machine in software defined network," *IEEE Access*, vol. 8, pp. 17404–17418, 2020.

[34] T. A. Pascoal, Y. G. Dantas, I. E. Fonseca, and V. Nigam, "Slow tcam exhaustion ddos attack," in *ICT Systerm Security Privacy Protection*, S. De Capitani di Vimercati and F. Martinelli, eds. Cham, Switzerland: Springer, 2017, pp. 17–31.

[35] Z. Li, H. Jin, D. Zou, and B. Yuan, "Exploring new opportunities to defeat low-rate DDoS attack in container-based cloud environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 695–706, Mar. 2020.

[36] N. Zhang, F. Jaafar, and Y. Malik, "Low-rate DoS attack detection using PSD based entropy and machine learning," in *Proc. 6th IEEE Int. Conf. Cyber Secur. Cloud Comput.*, Jun. 2019, pp. 59–62.

[37] N. Meti, D. G. Narayan, and V. P. Baligar, "Detection of distributed denial of service attacks using machine learning algorithms in software defined networks," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1366–1371.

[38] K. B. Virupakshar, M. Asundi, K. Channal, P. Shettar, S. Patil, and D. G. Narayan, "Distributed denial of service (DDoS) attacks detection system for OpenStack-based private cloud," *Procedia Comput. Sci.*, vol. 167, pp. 2297–2307, 2020.

[39] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, 1st Quart., 2014.

[40] F. Erlacher and F. Dressler, "FIXIDS: A high-speed signature-based flow intrusion detection system," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2018, pp. 1–8.

[41] R. Kumar and D. Sharma, "HyINT: Signature-anomaly intrusion detection system," in *Proc. 9th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2018, pp. 1–7.

[42] A. Patel, M. Taghavi, K. Bakhtiyari, and J. Celestino Jänior, "An intrusion detection and prevention system in cloud computing: A systematic review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 25–41, Jan. 2013.

[43] K. Vieira, A. Schulter, C. B. Westphall, and C. M. Westphall, "Intrusion detection for grid and cloud computing," *IT Prof.*, vol. 12, no. 4, pp. 38–43, Jul. 2010.

[44] S. L. Salzberg, "C4.5: Programs for machine learning by J. Ross Quinlan. Morgan kaufmann publishers, Inc., 1993," *Mach. Learn.*, vol. 16, no. 3, pp. 235–240, Sep. 1994.

[45] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[46] M. Wang, Y. Lu, and J. Qin, "A dynamic MLP-based DDoS attack detection method using feature selection and feedback," *Comput. Secur.*, vol. 88, Jan. 2020, Art. no. 101645.

[47] 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, 2018.

[48] A. Saied, R. E. Overill, and T. Radzik, "Detection of known and unknown DDoS attacks using artificial neural networks," *Neurocomputing*, vol. 172, pp. 385–393, Jan. 2016.

[49] C.-J. Hsieh and T.-Y. Chan, "Detection DDoS attacks based on neural-network using apache spark," in *Proc. Int. Conf. Appl. Syst. Innov. (ICASI)*, May 2016, pp. 1–4.

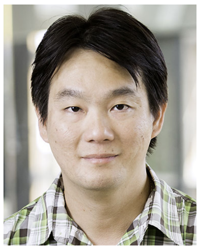
[50] T. Zhao, D. C.-T. Lo, and K. Qian, "A neural-network based DDoS detection system using Hadoop and HBase," in *Proc. IEEE IEEE 17th Int. Conf. High Perform. Comput. Commun.*, Aug. 2015, pp. 1326–1331.

[51] D. Perakovic, M. Perisa, I. Cvitic, and S. Husnjak, "Artificial neuron network implementation in detection and classification of DDoS traffic," in *Proc. 24th Telecommun. Forum (TELFOR)*, Nov. 2016, pp. 1–4.

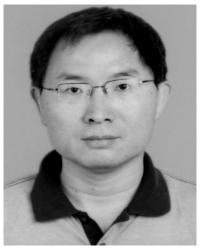
[52] R. M. A. Saad, M. Anbar, S. Manickam, and E. Alomari, ''An intelligent ICMPv6 DDoS flooding-attack detection framework (v6IIDS) using back-propagation neural network,'' *IETE Tech. Rev.*, vol. 33, no. 3, pp. 244–255, May 2016,
[53] S.-Y. Ji, B.-K. Jeong, S. Choi, and D. H. Jeong, ''A multi-level intrusion detection method for abnormal network behaviors,'' *J. Netw. Comput. Appl.*, vol. 62, pp. 9–17, Feb. 2016.
[54] G. S. Kushwah and S. T. Ali, ''Detecting DDoS attacks in cloud computing using ANN and black hole optimization,'' in *Proc. 2nd Int. Conf. Telecommun. Netw. (TEL-NET)*, Aug. 2017, pp. 1–5.
[55] D. E. Kim and M. Gofman, ''Comparison of shallow and deep neural networks for network intrusion detection,'' in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2018, pp. 204–208.
[56] (Jun. 2020). *CIC Dos Dataset (2017)*. [Online]. Available: https://www.unb.ca/cic/datasets/dos-dataset.html
[57] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, ''Botnet in DDoS attacks: Trends and challenges,'' *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2242–2270, 1st Quart., 2015.
[58] D. Arndt. (2015). *Flowtbag Program to Calculate Flow Statistics From a Given Capture File*. [Online]. Available: https://github.com/DanielArndt/flowtbag/wiki/features
[59] N. H. Oo and A. Htein Maw, ''Effective detection and mitigation of SYN flooding attack in SDN,'' in *Proc. 19th Int. Symp. Commun. Inf. Technol. (ISCIT)*, Sep. 2019, pp. 300–305.
[60] R. E. Kalman, ''A new approach to linear filtering and prediction problems,'' *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, Mar. 1960.

JESÚS ARTURO PÉREZ-DÍAZ received the B.Sc. degree in computer science from the Autonomous University of Aguascalientes, in 1995, for which he received the Best Student Award, and the Ph.D. degree in new advances in computer science systems from the Universidad de Oviedo, in 2000. He became a Full Associate Professor at the University of Oviedo, from 2000 to 2002. He was recognized by the COIMBRA group as one of the Best Young Latin-American Researchers, in 2006, and received a research stay at Louvain le nouveau University, Belgium. He has been awarded by the CIGRE and by Intel for the development of innovative systems. He is currently a Researcher and Professor with the ITESM–Campus Querétaro, México, and a member of the Mexican Researchers National System. His research interests include cyber security in SDN and design of communications protocols, where he has supervised several master and Ph.D. theses in the field.

ISMAEL AMEZCUA VALDOVINOS received the B.Sc. degree in computer science from the Universidad de Colima, in 2007, and the Ph.D. degree from the Tecnológico de Monterrey, Campus Cuernavaca, in 2013, where he worked on developing communication protocols for multi-homed devices. He is currently a Professor with the Facultad de Telemática, Universidad de Colima, México. His research interests include wireless sensor networks, Industrial Internet of Things (IIoT), and software-defined networks (SDN).



KIM-KWANG RAYMOND CHOO (Senior Member, IEEE) received the Ph.D. degree in information security from the Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). In 2015, he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He was a recipient of the 2019 IEEE Technical Committee on Scalable Computing (TCSC) Award for Excellence in Scalable Computing (Middle Career Researcher), the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, the Outstanding Associate Editor of 2018 for the IEEE Access, the British Computer Society's 2019 Wilkes Award Runner-up, the 2019 EURASIP JWCN Best Paper Award, the Korea Information Processing Society's JIPS Survey Paper Award (Gold) 2019, the IEEE Blockchain 2019 Outstanding Paper Award, the Inscrypt 2019 Best Student Paper Award, the IEEE TrustCom 2018 Best Paper Award, the ESORICS 2015 Best Research Paper Award, the 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, the Fulbright Scholarship, in 2009, the 2008 Australia Day Achievement Medallion, and the British Computer Society's Wilkes Award, in 2008.



DAKAI ZHU (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Pittsburgh, Pittsburgh, PA, USA, in 2004. He joined The University of Texas at San Antonio, in 2005, where he is currently a Professor with the Department of Computer Science. His current research interests include real-time embedded systems, low-power computing, fault-tolerant, and cloud computing. He was a recipient of the U.S. National Science Foundation Faculty Early Career Development Award in 2010.