IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 26, 2018, accepted February 15, 2019, date of publication March 15, 2019, date of current version April 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2904236

Cyberpulse: A Machine Learning Based Link Flooding Attack Mitigation System for Software Defined Networks

RAIHAN UR RASOOL[1], USMAN ASHRAF[2], KHANDAKAR AHMED[1], HUA WANG[1], WAJID RAFIQUE[3], AND ZAHID ANWAR[4,5]

[1]Centre for Applied Informatics (CAI), Institute of Sustainable Industries and Liveable Cities, Victoria University, Footscray, VIC 3011, Australia
[2]Department of Computer Networks and Communications, King Faisal University, Hofuf 31982, Saudi Arabia
[3]Department of Computer Science and Technology, Nanjing University, Nanjing 210008, China
[4]School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad 44000, Pakistan
[5]Mathematics and Computer Science Department, Fontbonne University, St. Louis, MO 63105, USA

Corresponding author: Raihan Ur Rasool (raihan.rasool@live.vu.edu.au)

This work was supported by the Deanship of Scientific Research (DSR) at King Faisal University under Grant 186147.

ABSTRACT Software-defined networking (SDN) offers a novel paradigm for effective network management by decoupling the control plane from the data plane thereby allowing a high level of manageability and programmability. However, the notion of a centralized controller becomes a bottleneck by opening up a host of vulnerabilities to various types of attacks. One of the most harmful, stealthy, and easy to launch attacks against networked systems is the link flooding attack (LFA). In this paper, we demonstrate the vulnerability of the SDN control layer to LFA and how the attack strategy differs when targeting traditional networks which primarily involves attacking the links directly. In LFA, the attacker employs bots to surreptitiously send low rate legitimate traffic on the control channel which ultimately results in disconnecting control plane from the data plane. Mitigating LFA on the control channel remains a challenge in the network security paradigm with the use of network traffic filtering only. To address this challenge, we propose CyberPulse, a novel effective countermeasure, underpinning a machine learning-based classifier to alleviate LFA in SDN. CyberPulse performs network surveillance by classifying network traffic using deep learning techniques and is implemented as an extension module in the Floodlight controller. CyberPulse was evaluated for its accuracy, false positive rate, and effectiveness as compared to competing approaches on realistic networks generated using Mininet. The results show that CyberPulse can classify malicious flows with high accuracy and mitigate them effectively.

INDEX TERMS Link flooding attacks, SDN security, OpenFlow, deep learning.

I. INTRODUCTION

Software Defined Networking has been proposed in the wake of increasing network scale and management complexities due to the continuous development in current networks. SDN reduces network complexities by providing a simplified, flexible, dynamic, and centralized network management using the concept of separate layering for data and control planes [1]–[3]. In SDN environment, the data plane is responsible for providing traffic forwarding functionality while a centralized controller maintains a global view of the entire network and can be easily programmed for the desired traffic forwarding [4], [5]. SDN utilizes the OpenFlow (OF) protocol for communication between controller and data plane infrastructure [6]. Every OF-enabled device in the data plane has flow tables that are managed by the controller which contain the entries called flow rules to route the traffic to its destined path. All incoming packets are compared with the entries in flow table, if, flow entries are not found for a specific flow, a control packet is sent to the controller to request further action [7]. The controller then updates the flow rules after the packet inspection and traffic is forwarded towards the destination. There is a continuous interaction between the controller and the data plane for traffic forwarding. However, this continuous communication can also lead to serious security issues if disrupted. A skilled adversary can exploit this security challenge by causing DoS and disrupt this communication [8], [9]. A more serious threat is

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

2169-3536 © 2019 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

when the control channel is attacked by creating congestion on the link with anomalous traffic using a Link Flooding Attack (LFA) [10].

As the name implies, LFA is a link-based attack, where the link connecting to a target server is flooded to cause traffic congestion which ultimately disrupts the legitimate traffic to the target server. Initially, the attacker identifies a target server and creates a link map around it by sending traceroute commands [11]. Subsequently, specific hosts called decoy servers are selected that are placed around the path of the target link. After identifying the server, a set of bots is selected which can generate sufficient traffic to flood the link. Finally, the decoy servers are manipulated by these bots to send low rate traffic to each other in order to cause congestion on the target link which can ultimately disrupt communication of the target server with the rest of the network.

Some studies have been performed recently that attempt to mitigate LFA in SDN. A few of the available studies focus on exposing a fake network topology map to the adversaries [12]–[14]. Some other techniques are based on link inspection which performs a similar operation but on the basis of critical links supervision [10], [12], [15]. Most of the available literature does not clearly distinguish the attack behavior on SDN versus traditional networks [15]–[19]. While some authors have used SDN test beds to perform the experiments [20], [21] or SDN based mitigation techniques there remains a lack of literature available that mitigates LFA on SDN, specifically control channel attacks. The critical reason behind the failure of traditional techniques in SDN context is that SDN employs centralized control strategy to handle the network traffic. The challenge is that the control channel is not directly accessible to the normal traffic. Therefore, a attacker needs to perform specially crafted packet misses to attack this channel. For the same reason, mitigating LFA on control channel is a complex task.

CyberPulse is a novel solution because it taps into machine learning and artificial neural networks, a type of artificial intelligence to select appropriate traffic features for accurate classification in a large volume of traffic data. Machine Learning (ML) allows machines to learn about the features of a problem using statistical techniques and automate the solution for an arbitrary dataset. ML algorithms today are being used to predict stock prices, weather patterns, highway traffic, and have been successfully used for intrusion detection systems as well. This motivated us to employ machine learning for the LFA problem. Our results show that it is effective into addressing the challenge of mitigating control channel LFA.

In this research, we start by highlighting the problem and use extensive experimentation to show how LFA poses a threat to modern SDNs. Our goal is to formulate the LFA problem as a machine learning problem and tap into existing knowledge base of state-of-the art algorithms developed by this community to develop a classifier that allows us to achieve high accuracy. We therefore detail how we sought machine learning algorithms and performed feature selection for a deep-learning solution suitable to our problem. We then outline the design of our testbed using the Mininet network emulator [21] and Floodlight [22] web-based controller version 1.2. We chose to use Mininet for our evaluation testbed because it is widely recognized as realistic emulator for deploying large networks particularly SDN. It provides performance accuracy and scalability and is a preferred option as opposed to simulators and shared hardware testbeds. We achieve flooding on the network with a variable number of attackers and show the effect of attacks on SDN. CyberPulse is implemented as an application at the application layer of the SDN controller. We propose the following contributions:

- The challenge of LFA on SDN control channel is highlighted with the help of extensive experiments. We show how LFAs can degrade the performance of SDN and if no precautionary measures are performed, how these attacks can bring down the entire network.
- CyberPulse, a novel solution for detecting and mitigating LFA on the SDN control channel is proposed by leveraging the application layer of the SDN controller.
- Flood traffic classification is performed using deep learning-based techniques on state of the art real world cyber risk research and decision support dataset from the UCI machine learning repository.
- A comprehensive evaluation has been conducted to assess the performance of deep learning classification and a side-by-side comparison has been made with competing techniques.

The rest of the paper is organized as follows, Section II discusses system adversary model whereas Section III illustrates the machine learning approach and algorithm selection strategy. Section IV describes the detailed architecture of CyberPulse, in the same way, Section V illustrates multilayer perception in classifying LFA traffic. Section VI provides the implementation details of CyberPulse, Section VII presents the related work, and finally, Section VIII concludes the paper.

II. SYSTEM AND ADVERSARY MODEL

In this section, we present LFA adversary model used in this research, we first describe the list of acronyms used in this research in Table 1. Initially, we discuss about SDN flow rule installation, a flow rule instructs the switch on how to handle an incoming traffic packet, an OF switch contains flow entries that consists of priority, match, timeout, and instruction fields. Flow entries in an incoming packet at the OF switch are matched with the flow table in order to forward the traffic. When a corresponding flow rule for the incoming packets is found, the received and byte counters are incremented and the flow is handled according to the entries in the flow table. In case a flow rule is not found for an incoming packet, a PACKET_IN message containing header information of the packet is forwarded to the controller for further instructions on how to handle the packet. The controller parses the header fields of the packet and sends a PACKET_OUT message

TABLE 1. List of acronyms.

Symbol	Description	Symbol	Description
LFA	Link Flooding Attack	ANN	Artificial Neural Networks
SDN	Software Defined Network	ML	Machine Learning
IP	Internet Protocol	TCP	Transmission Control Protocol
ISP	Internet Service Provider	MLP	Multi-Layer Perception
DoS	Denial of Service	ML	Machine Learning
LAN	Local Area Network	TCP	Transmission Control Protocol
VM	Virtual Machine	REST	Representational State Transfer
API	Application Programming Interface	D4J	Deep Learning4J
ASes	Autonomous Systems	VN	Virtual Networks
IDS	Intrusion Detection system	CoDef	Collaborative Defense
OF	OpenFlow	VPN	Virtual Private Network
DNS	Domain Name System	IPS	Intrusion Prevention System
BHP	Burst Header Packet		

containing flow rule to the OF switch which subsequently, forwards the packet and update the flow table.

A. TAXONOMY OF LFA

In LFA, adversaries send low rate legitimate flows to precisely selected servers around the target link called decoy servers. This results in an increase in traffic on the target link, as the number of flows increases. Initially, the network traffic tends to slow down and with the passage of time when the link is severely flooded, the target server becomes irresponsive. In Fig. 1 normal SDN traffic flow is shown, it can be observed that when a packet arrives on a switch, a request is generated to the controller for path selection by inspecting the packet header. Subsequently, the relevant rules are forwarded to the switch and corresponding packet is transferred towards the destination. LFA mimics the same normal activity of the network and utilizes its low rate nature to avoid detection and flood the entire network.

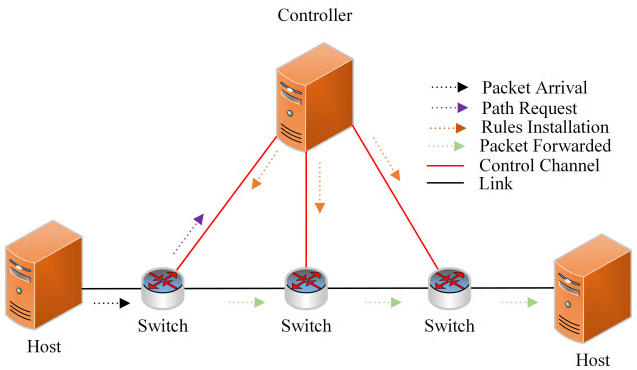

FIGURE 1. Normal SDN operation.

Attackers use low rate traffic which is hard to detect. They keep on sending traffic to the decoy servers until the link is congested with the attack traffic. Fig. 2 shows an example

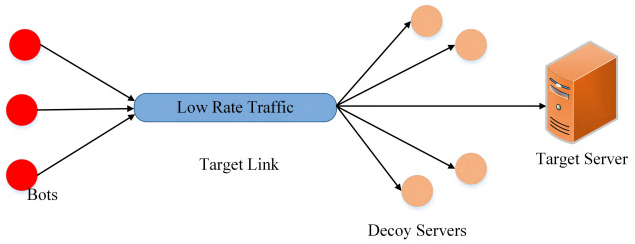

FIGURE 2. LFA model where bots send traffic to decoy servers and congest target link. (Redrawn and extended the figure at [19].)

of LFA on a selected link, it can be observed that three bots are sending traffic to 4 decoy servers which is passing through the target link. A server is attached to the target link, so the low rate traffic will occupy the link bandwidth and ultimately, obstruct the legitimate traffic towards the target server. To increase the effectiveness of the attack, a high number of bots are manipulated to send traffic to the decoy servers.

B. LFA ADVERSARY MODEL

The adversary seeks to disconnect the data plane from the control plane by employing the link flooding technique. The attacker first constructs a link map of the network by utilizing layer 3 diagnosis tools such as traceroute commands targeting different points in the network [23]. The information gathering host is called a link prober and the identified link map describes the routing policy towards the target area in the SDN. The adversary figures out the best attack-cost strategy and selects the links which can be occupied by as many bots as possible to send increasing amount of attack traffic. Afterwards, it utilizes bots to sends TCP like traffic to the decoy servers. It is also pertinent to note that the adversary tries to send traffic at a low rate scale to avoid being detected by the rate limit detection mechanisms. Subsequently, the link capacity is fully utilized by the attack traffic, which will

impede the legitimate traffic to flow through the link and ultimately, connection to the victim server will be subverted.

In SDN the control channel can be attacked by flooding data plane switches with flood packets which trigger new flow rule installation. The buffer memory of the switch will be full and it will encapsulate the whole packet in the PACKET_IN message which will result in flooding on the control channel and increasing the latency of flow rule installation. Furthermore, in extreme conditions, the control channel can be disrupted from the rest of the network. This attack approach is applicable to backbone SDN networks that are increasingly being employed to ISPs across the world who want to benefit from the performance gains as well as ease of maintenance compared to traditional networks. Some previous studies have also explored the vulnerable nature of SDN under attacks [24]–[26].

III. MACHINE LEARNING APPROACH AND ALGORITHM SELECTION

There are several well-established ML techniques, algorithms, and tools available and there is no distinct best candidate for a specific task. The particular selection however, affects and usually involves a trade-off between several factors including but not limited to prediction accuracy, performance, and the training dataset size depending on the nature of the problem. In this section, we discuss how we gave considerable thought to this decision by sharing our comparative review of the state-of-the-art machine learning tools currently available and how they apply to our problem. Table 2 contains the comparison of different machine learning algorithms based on training time and accuracy.

TABLE 2. Comparison of machine learning algorithms for classification.

Algorithm	Training Time	Classification Accuracy
Multiclass logistic regression	Fast	Low
Multiclass neural networks	Slow	High
Multiclass decision forest	Fast	Low
Multiclass decision jungle	Slow	High
Two class decision forest	Fast	Low
Two class boosted decision tree	Fast	Low
Two class decision jungle	Slow	Low
Two class SVM	Slow	High

A. DRAWING INSPIRATION FROM MACHINE LEARNING FOR SDN SECURITY ANALYSIS

We draw CyberPulse inspiration by observing that in SDN the controller has access to a large volume of important traffic statistics that may be collected at specific time intervals. Specifically, statistics regarding the individual flows are made available by OF switches. The precise time interval for statistics collection is an important determining factor where a small interval increases overhead, similarly, a large interval increases the detection time. A machine learning classifier can then be utilized to help the classifier identify the flood traffic source. As a first step, a classifier is developed where an algorithm is used to build the classifier model using the provided training dataset. The model is then employed for the classification where the accuracy is determined by the percentage of test data records that are correctly classified.

B. MACHINE LEARNING TOOL SELECTION

For our research, we preferred having several key features in our ML tool of choice. These features in decreasing order of priority are as follows: (1) freely available, (2) classification accuracy, (3) better performance for medium-sized datasets, availability of documentation, and portability. In this regard, we conducted a comprehensive survey of the ML tools available in the market and examined the pros and cons. In Table 3, we have compared WEKA, D4J (Deep Learning4J), Tensor Flows, and Encog3 based on the following parameters: source code, help and support, license type, programming language support, compatibility, documentation, and performance. Details description of comparison parameters is given below:

1) **Source Code:** It corresponds to the underlying language used for the development of the machine learning tool.
2) **Help & Support:** Help correspond to the publicly available assistance online and support refers to the help provided by the machine learning tool provider himself.
3) **Performance:** We have compared performance of the tools based on the training time taken to perform the classification task, on a scale of 0 through 10 with 10 being the maximum.
4) **Documentation:** Availability of documentation, guides developers to use the tool in the required circumstances, it has been rated in a range from 0 to 10 with 10 corresponds to a maximum score.
5) **Programming Language:** Over the period of time, new programming languages are developed hence, supported programming languages play an important role in the selection of a machine learning tool.
6) **Compatibility:** Sometimes models developed using an older version of a programming language, are not supported by the newer version of the available tool. A good tool supports backward compatibility.
7) **Dataset Size:** Certain ML tools tend to support big data processing at reasonable performance depending on the design and the models used.
8) **Development Mode:** Some ML tools tend to have a steeper learning curve and experience required for model development as compared to the others.

Encog3 is a CSharp based framework that can accommodate a range of ML techniques for medium sized datasets. Support is however limited and the learning curve is steep. TensorFlow is based on Python and also supports a range of ML techniques. In addition, it is suitable for large datasets

TABLE 3. A comparison of state of the art machine learning tools used in research.

Comparison Parameters	Weka	D4J	TensorFlow	Encog3	PyTorch
Source Framework	Java	Java	CPP engine	CSharp	Lua-based
License	Opensource	Opensource	Opensource	Opensource	Opensource
Programming Language	Java	Java	Python	CSharp	Python
Compatibility	No issues	No issues	Less backward compatible	Less backward compatible	Less backward compatible
Dataset Size	Small datasets	Larger datasets	Big datasets	Medium sized datasets	Big datasets
Development Mode	GUI-based	Code-based	Code-based	Code-based	Code-based
Help and Support	9	6	7	5	4
Documentation	9	6	7	5	6
Performance	9	7	6	8	9

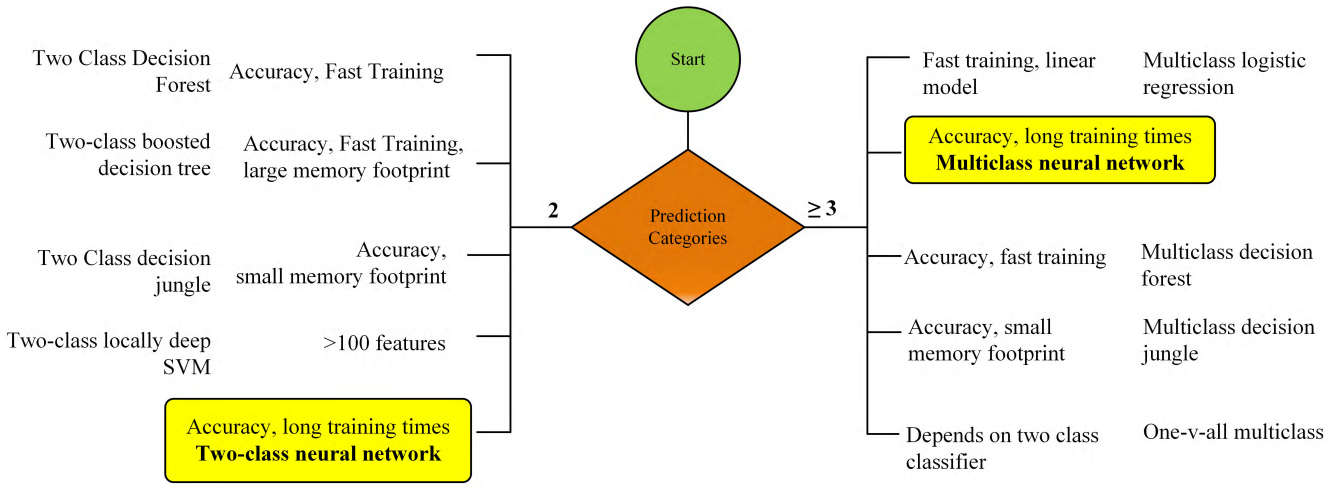

FIGURE 3. Machine learning algorithm selection for LFA classification.

and the computational accuracy is also very high. The model development process involves handling graph data structures which can be time-consuming. The learning curve is also comparatively high. D4j is a java-based machine learning tool which requires high-performance computing machines for a successful operation. It supports all the available machine learning tasks and is suitable for medium and large sized datasets. PyTorch is a lua-based deep learning framework, it works on the basis of tensor which considers every model as a directed acyclic graph. In TensorFlow the graph is statically defined before a model can run and the input is extracted from the outer world by *tf .Session* and *tf .Placeholder* interfaces. On the other hand, PyTorch is more dynamic where nodes of the directed acyclic graph can be defined, changed, and executed dynamically. No session and placeholder interfaces are explicitly required. This framework is tightly coupled with python. WEKA is based on Java and supports classification and clustering. It works efficiently for small to medium sized datasets and includes a rich set of ML algorithms. WEKA has extensive help and support available and supports models developed using older versions of the tool.

In summary, WEKA is suitable for small datasets classification and clustering tasks. Encog3 can be utilized machine learning tasks on medium sized datasets. D4j and TensorFlow can be deployed for big datasets and complex machine learning tasks. WEKA seemed to be the closest to our requirements and therefore we chose it for our implementation.

C. MACHINE LEARNING ALGORITHM SELECTION

Numerous algorithms are available to choose from when we come across a ML classification problem. The selection of an algorithm depends on the type of problem we are dealing with, however, the algorithmic performance depends on the size and the structure of data. Fig. 3 shows the classification algorithm selection strategy. Initially, when an ML task is assigned, the first question to ask is that how many numbers of classes is to be predicted, if there are two classes, then further decision depends on multiple aspects i.e. accuracy, training time, and performance. In the next step, the analysis is performed to assess accuracy and training time. In CyberPulse, we select the ANN technique as the foremost priority for flooding attacks classification was accuracy because if the malicious traffic can be accurately classified, then it can be easily mitigated.

Table 2 further compares different ML algorithms based on their training time and classification accuracy.

Fig. 3 shows two class and multi-class categories in the start. In the multi-class, classification categories, both neural networks and decision forests perform classification tasks with fast training time and high accuracy. Multi-class and two class neural networks consume more training time but yield good classification results. Therefore we select artificial neural networks for flooding attack traffic classification because the classification accuracy was of more importance than the training time. In the Table 2, the performance attributes have been assigned as either low, high, slow, or fast.

IV. CYBERPULSE ARCHITECTURE AND DESIGN

To address the problem discussed in section III, we propose CyberPulse an extension application module in the SDN controller to secure SDN against LFA. While the CyberPulse design is generic, our prototype implementation resides on application plane of the Floodlight controller and performs the LFA mitigation process. We, therefore, refer to Floodlight when we discuss the design of the controller in the rest of this document. The prime purpose of CyberPulse is to detect and eliminate LFA on the control channel. In this section, we, discuss the detailed architecture of CyberPulse.

A. OVERALL CYBERPULSE ARCHITECTURE

CyberPulse exploits the northbound REST API of the SDN controller [22] to communicate with the controller and perform the operation. As shown in Fig. 4, CyberPulse is an extension module in the application layer of the SDN controller. It works concurrently with other modules of SDN in order to provide the required functionality. CyberPulse incorporates three modules which includes Link Listener, Flood Detection, and Flood Mitigation. CyberPulse uses the REST API to connect with the controller and in-turn the controller uses Southbound OF API to communicate with the data plane switches [27].

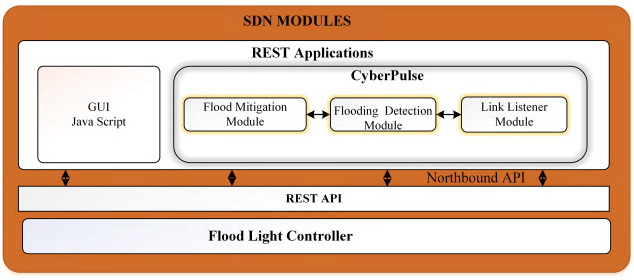

FIGURE 4. The overall architecture of CyberPulse showing CyberPulse modules.

B. CYBERPULSE MODULES

It can be observed from Fig. 4 that there are three modules in CyberPulse. Each module performs a specific operation in order to accomplish the cumulative task of LFA mitigation. CyberPulse only requires modifications in the software and no hardware is required for its operation. It can, therefore, be easily integrated with the existing Floodlight controller. CyberPulse modules are explained here:

1) LINK LISTENER MODULE

The process starts with the Link Listener module which continuously inspects the control channel and constantly provides traffic flow statistics to the Flood Detection module. Table 4 shows the traffic statistics that can be extracted by the Link Listener. Some statistics are directly extracted, however, some of them are calculated based on other statistics. Floodlight exposes a Java-based REST API to extract network statistics, which is employed by this module to retrieve the required statistics.

2) FLOOD DETECTION MODULE

Flood Detection module inspects the statistics and performs flow classification. This module incorporates the statistics pre-processing component which eliminates the packet headers information such as ack, $syn-ack$ packets from the statistics and presents only traffic flows to the Flood Detection module. These statistics are consumed as features by a deep learning sub-module that uses Artificial Neural Networks (ANN) a type of ML algorithm to classify the network traffic. The steps involved in the Flood Detection module are given in Algorithm 1.

Algorithm 1 LFA Classification Using MLP

Require: Traffic Flow F, packet statistics, dataset T
Ensure: Flow Class: *Flooding*, *Legitimate*

```
Get traffic flows
for Flow f ∈ F do
    Get Flow Statistics
    Extract Features
end for
Pre-process T
Train ANN-MLP model using T
Classify flows using ANN-MLP
Export classification
return Flow Class: Flooding, Legitimate
```

The ANN classification module builds a training model by utilizing flooding attack datasets. Subsequently, based on this trained model, it performs the classification process. The classifier outputs the benign and the attack flows.

3) FLOOD MITIGATION MODULE

Results of the classification are forwarded to the Flood Mitigation module which drops the attack flows using the null routing technique. The operation of CyberPulse is explained in the Algorithm 1. A Null route creates a block-hole, which is a kernel routing table entry leading to nowhere. Packets matching a null route will be dropped.

V. MULTILAYER PERCEPTION IN CLASSIFYING LFA TRAFFIC

This section delves into the detailed design of the deep learning based nonlinear Multilayer Perception (MLP) backward propagation structure used for network traffic classification. The basic computation units of the MLP are illustrated in Fig. 5. A node calculates the weighted sum of inputs, incorporates a node threshold and forwards the results to a nonlinear function. MLP is characterized by an input, an output, and one or more hidden layers that are interlinked with each other. Oppositely, there is no connection between the nodes in the same layer, moreover, there are no bridging connections either. The role of the backward propagation and the hidden layers in MLP are discussed next.

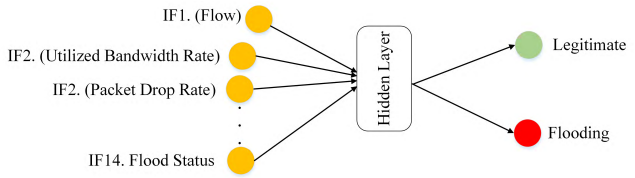

FIGURE 5. The MLP configuration used in the experiment.

1) **Backward Propagation:** The backward propagation process in this algorithm is used to continuously adjust weights of the layers in order to minimize the difference between actual output and the desired output. It is used to calculate a gradient descent which is subsequently used to calculate weights and finally used to train the MLP model.
2) **Hidden Layers:** Hidden layers are the neuron nodes that reside between inputs and the outputs, which correspond to the number of intermediate connections that are used in the experiment, and by default this number is selected as 1. The number of hidden layers depends on the mean value of the input and output layers. When data is linearly separable, the number of hidden layers to use is 0. These layers actually transform the single layer perception into multilayer which is denoted by $h : (h_1, h_2, \ldots, h_n)$. Where hidden layer comprised up of 1 to a maximum number of n layers. Each neuron has an associated weight, which is its contribution towards the actual output of the classification.
3) **Output Layer:** Output layer is the number of possible classes in the dataset that we want to predict and is denoted by y. In our experiment, this is equal to two for *Legitimate* and *Flooding*. W is the weight learned from the training set by iteratively minimizing the error using gradient descent as can be seen in the equation 1. The gradient used in the equation can be determined by using the backward propagation algorithm.

$$W_{next} = W + \Delta W \tag{1}$$

$$\Delta W = -learning\ rate \times gradient \times momentum \times W_{previous} \tag{2}$$

The change in weight ΔW can be calculated by employing the equation 2 which multiplies gradient decent by the learning rate and adds the previous change in weight $W_{previous}$. The input layer is associated with the number of attributes in the dataset and it is denoted with x. Input dataset is designated as the $(x_1, x_2, \ldots, x_m)$ features having influence on the final output of the flood traffic classification where subscripts 1 to m denotes the number of features of the input layer. The reason behind using notation of n for hidden layers and m for input layers was that, the number of hidden layers neurons may differ from the number of input data. If we have m input data features, we multiply it with weights $(w_1, w_2, \ldots, w_m)$ we get the equation 3.

$$W.X = w_1x_1 + w_2x_2 + \ldots + w_mx_m. \tag{3}$$

Subsequently, the input dataset is multiplied with the hidden layer neurons weights to get the result $z(w_1^1, w_2^1, \ldots, w_m^1)$, $z = \sum_{i=1}^{m} w_i x_i + bias$ adding this value to the activation function $f(z)$ to get the output of the first hidden neuron for the whole dataset. The superscript 1 denotes the current hidden layer, if there are multiple hidden layers then the superscript is used to identify the hidden layer number. In this example, all the weights have superscript 1 which shows that they are weights of the hidden layer 1. This process is applied to all the features of the training dataset. For the final output, a dot product of hidden layer output and hidden layer weights wh is performed. The set of weights consist of $w_1h : (w_1^{h1}, w_2^{h1}, \ldots, w_n^{h1})$, with n weights as n corresponds to n hidden layer inputs. Finally, a *bias* value is added to obtain the result and is given in the equation 4 similarly equation 5 contains the final output of the MLP algorithm. The value of z is fed to an activation function to get the output for the output layer.

$$z = \sum_{i=1}^{n} w_{i=1}^{h1} \times h_i^1 + bias \tag{4}$$

$$r\hat{y} = f(z) \tag{5}$$

VI. IMPLEMENTATION DETAILS

In this section, we describe how we conducted simulations and analysis to evaluate the performance of CyberPulse in two conditions including flooding attack and in normal network operation scenario. As a way of discussion we build a case based on our evaluation results that while the traditional SDN environment offers no flooding attack defense mechanism, CyberPulse bridges that gap and is an effective extension to the SDN environment. Fig. 6 represents an example how we use some hosts as adversaries to simulate an actual LFA environment. This example is based on the adversary model outlined earlier targeting the control channel. In this figure, we can observe a small tree topology consisting of 8 hosts where the adversaries are sending traffic towards the target link to flood it and obstruct the legitimate traffic. By using this strategy, any link in the network can be congested which will result in causing legitimate traffic delays. Fig. 7 shows

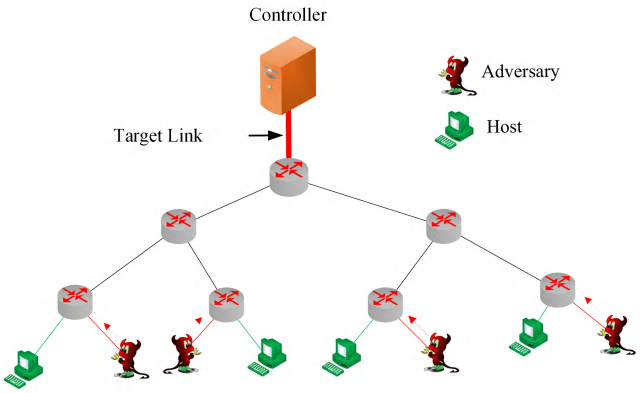

FIGURE 6. An example of adversaries sending attack traffic.

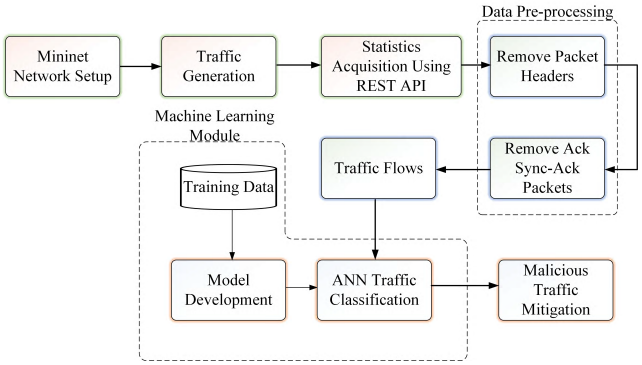

FIGURE 7. Flow diagram of CyberPulse operation.

the flow diagram of CyberPulse process, it starts by setting-up a Mininet network and generating network traffic. Subsequently, statistics of switches and ports are extracted using REST API. Data preprocessing is performed to extract only data packets removing extra header packet information, then machine learning module performs the traffic classification and identifies the malicious flows. Finally, the malicious flows are mitigated using null routing technique.

A. EXPERIMENTAL SETUP

As it has been discussed in Section IV that CyberPulse consists of three modules, we study the working principle of all of these modules in this section with the help of experimental analysis. A virtual network is designed using a single desktop computer to implement CyberPulse in an emulated network environment. In our case the computer specifications were as follows: Intel(R) Xeon(R) CPU E3-1225 v5 @ 3.30GHz and 16 GB RAM. Windows 10 was running on the host machine and Ubuntu 14.0.4 was running on an Oracle VM Virtualbox. We used Floodlight open-source controller, programmed in the Java language. In this section initially, we present the parameters for network creation and traffic generation. The Link Listener module consists of a statistics collector which performs the network surveillance and collects statistics. In our experiment, we use Wireshark to get network statistics which are classified by the Flood Detector module. Subsequently, we present the deep learning-based MLP parameters used in the experiment. Algorithm 1 explains the classification procedure, the algorithm takes input of traffic flows set F and training dataset (T). The algorithm gets the traffic flows and for each flow, traffic statistics are computed, after the extraction of all the features, the preprocessing is performed. Subsequently, the classifier is trained using ANN-MLP algorithm, finally, the trained model is utilized to classify the captured network statistics into final classes as *Flooding* or *Legitimate*.

The dataset used in our research was downloaded from the UCI machine learning repository for Burst Header Packet (BHP) flooding attacks which consisted of 22 feature [28]. After careful investigation of the dataset and proposed problem, we selected 14 features for the training and testing of the traffic. The features information of the dataset is given in Table 4. The behavior of each flow is inspected and a point is identified where it was misbehaving. The MLP classifier model performs its operation by classifying the incoming traffic flows into possible classes, i.e. *Legitimate* and *Flooding*. It is worth mentioning here that the training set consisted of *Legitimate* and *Flooding* class flows, 88% instances related to *Legitimate* and 12% to *Flooding*.

TABLE 4. Features used in the research and the statistics collected.

Sr. No	Feature Name	Sr. No	Feature Name
1	Node	8	Used Bandwidth
2	Utilized Bandwidth	9	Lost Bandwidth
3	Packet Drop Rate	10	Packet Size
4	Full Bandwidth	11	Packets Received
5	Percentage of packet lost Rate	12	Packets Lost
6	Percentage of lost byte rate	13	Transmitted Byte
7	Packets received rate	14	Flooding Status

Previous studies [29], [30] suggest that a utilization ratio of more than 40% indicates network performance degradation. In the same way, link utilization percentage is another indicator of possible threats to the network. Node utilization is calculated using equation 6.

$$utilization(\%) = \frac{datasetsize}{Bandwidth \times interval} \times 100 \qquad (6)$$

In the experiment, the number of input neurons was 14 and the output neurons were equal to 2 corresponding to the desired output of the experiment i.e. *Flooding* and *Legitimate*. The model building parameters are given in Table 5. The training time was set to 500 secs having a training and test set size of 60% and 40% respectively which was later changed in the data split experiment. The experiment was performed using 10 fold cross-validation. In experiment 8 hidden neurons were used which is the average of the input and output layers. The learning rate was set to 0.3 as on this learning rate the algorithm was behaving efficiently in terms of accuracy and

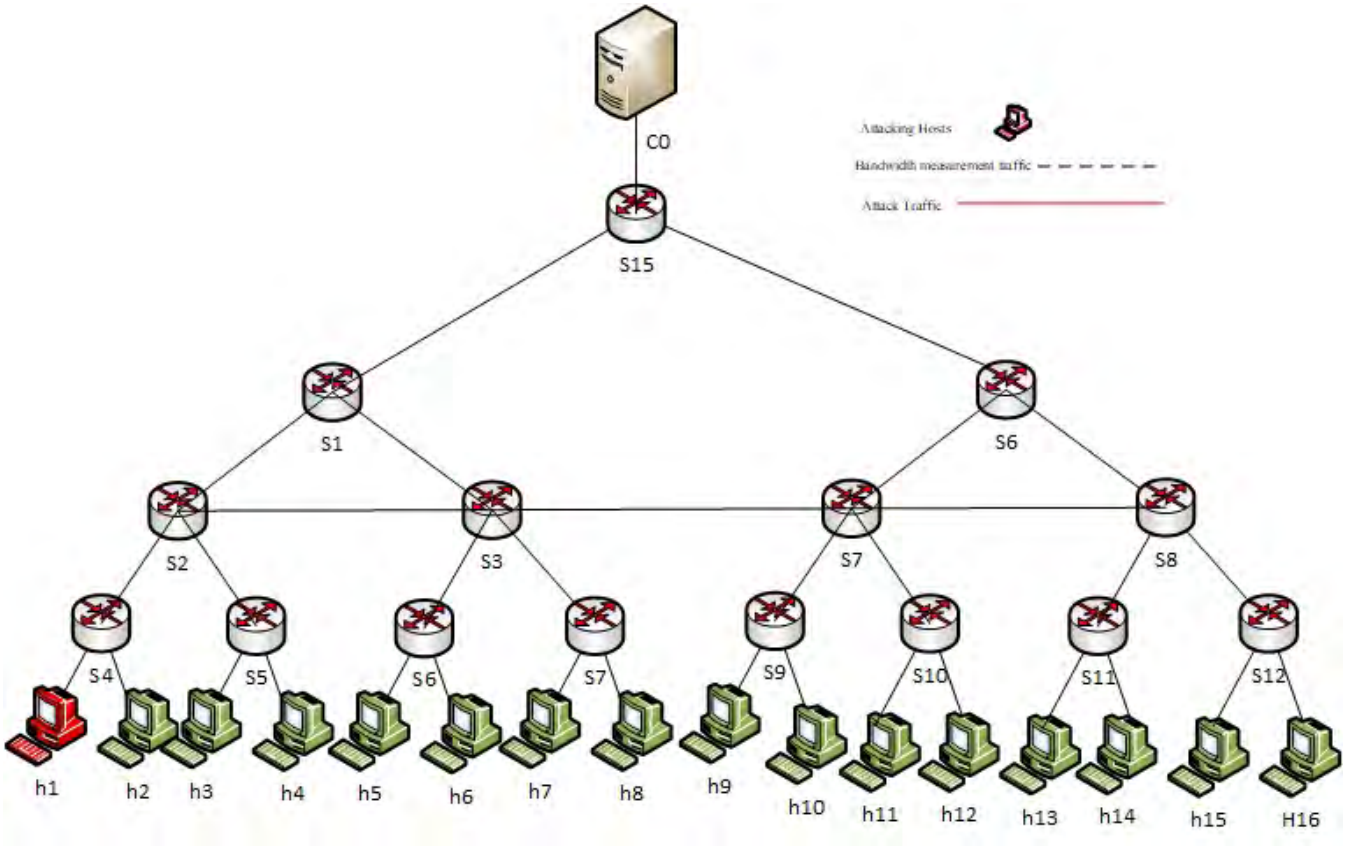

FIGURE 8. Emulated topology using Mininet and Floodlight controller.

TABLE 5. Model building parameters and their values.

Model parameters	Value
Input Neurons	14
Output Neurons	2
Hidden Neurons	8
Training time	500Sec
Actual time for model building	17.4Sec
Learning Rate	0.3
Validation	Cross 10 Fold
Momentum	0.2

time constraint. The experiment was performed using one hidden layer and a momentum of 0.2.

The network topology was designed as shown in the Fig. 8. The figure represents a tree topology with three levels consisting of 16 hosts and 12 OF switches. The reason behind using this topology is that it is always a cost-security trade-off when employing a network topology, since installing an out-of-band channel may constitute significant infrastructure cost if the SDN controller needs to be connected to every available switch. SDNs typically used in large scale ISPs may span few hundreds switches which can pose a significant cost. This topology is created in Mininet running on the Ubuntu operating system. The remote controller running on Windows 10 machine is utilized and accessed from the Ubuntu machine using port 6634. The *iperf* tool is used to send flood as well as legitimate traffic to multiple hosts in the network and plot graphs to show the effect of flood traffic on the links. The emulated network topology consisted of the parameters given in Table 6.

TABLE 6. Parameters for topology creation.

Parameter	Value
Virtual Hosts	16
OF Switches	12
Controller Port No	6653
Link Bandwidth	1000Mbps
Mininet OS	Ubuntu
Controller OS	Windows 10
Packet Capture	Wireshark

1) FLOOD TRAFFIC CAPTURING

The test-bed network comprised of 12 OF switches, and 16 hosts, connected with the controller each having a link capacity of 1000 Mbps as shown in the Fig 8. To present the LFA attack model in the experiment bots and decoy

servers are deliberately given the roles utilizing the available 16 hosts. Subsequently, low rate traffic is generated towards target hosts. *Iperf* is an open source tool to create and measure network traffic. We use *iperf* $-s$ command to start a TCP server connection. Using this command the server will start listening on TCP port 5001. Subsequently, any host can connect to the server using the command: *iperf* $-s$ $-c$ 10.0.0.1 $-t$ 15, which specifies the IP address of the server to connect and time duration for which the connection is requested. At the end of connection a summary of data transfer and bandwidth is displayed on both client and server terminals. After a TCP connection completion, the server keeps on listening to the port creating an opportunity for any other host to connect with the server.

Experiments are performed by sending flood traffic towards the target servers and analyzing the link bandwidth. The following Mininet command is used to create a tree topology and fixing the bandwidth of the links to 1000 Mbps.

$sudomn - -controller = remote,$
$ip < ip\ of\ windows\ 7\ machine : port >$
$- - topo = tree, 3\ link\ tc,\ bw = 1000$

Traffic is manipulated by opening a separate terminal for every attack node in the Mininet Ubuntu machine. During the flows generation, Wireshark [31] network packet statistics analysis tool has been utilized.

2) ATTACK SIMULATION

To simulate a real world LFA scenario, we manipulate different hosts in the network as bots to send low rate traffic to attack the OF channel. Bandwidth of the links is measured before and after sending the flood traffic. Hence, for instance in one experiment H2 was acting as a decoy server and H3, H4, and H5 were behaving as bots. In each experiment, the number of decoy-bot pairs have been varied and bandwidth consumption of the links is measured. The experiment was performed with one, two, three, and up to 14 attackers and all the traffic statistics given in Table 4 of the traffic are measured.

The experiment was run multiple times and each time the reported bandwidth was extracted using Wireshark. With the increase in the number of attacking hosts, the bandwidth tend to decrease. Fig. 9 shows the effect of LFA on available bandwidth and packet drop rate in the network. Fig. 9a shows the effect on available bandwidth with the increase in the number of attackers. It can be observed from Fig. 9a that with the increase of number of attackers, the bandwidth started to saturate and when the number of attackers reached 14 the available bandwidth reduced to nearly a mere 50 Mbps, which demonstrates the devastating effect of LFA on the SDN control channel. Another phenomena was noted that with the increase in the number of attackers the packet loss rate also tends to increase as given in the Fig. 9b. It can be observed that when the attack traffic was increased the packets drop rate also increased. There was a rapid vertical shift in the packet drop rate when the number of attackers increased to 14, which shows that when the number of attackers has been increased to a certain level, the packet drop rate increases exponentially. After running the experiment, the flows were extracted for every node in the network. After extracting all the traffic, data preprocessing was performed and statistics for the individual flows were captured. Subsequently, the training set was used for building the model and subsequently, the trained model was used to classify the statistics into benign and malicious flows.

B. RESULTS

In this section, we discuss and analyze the results for the experiments. The results are based on two scenarios: normal SDN traffic flow and during the attack. According to previous studies [29], [30] when the utilization ratio of the

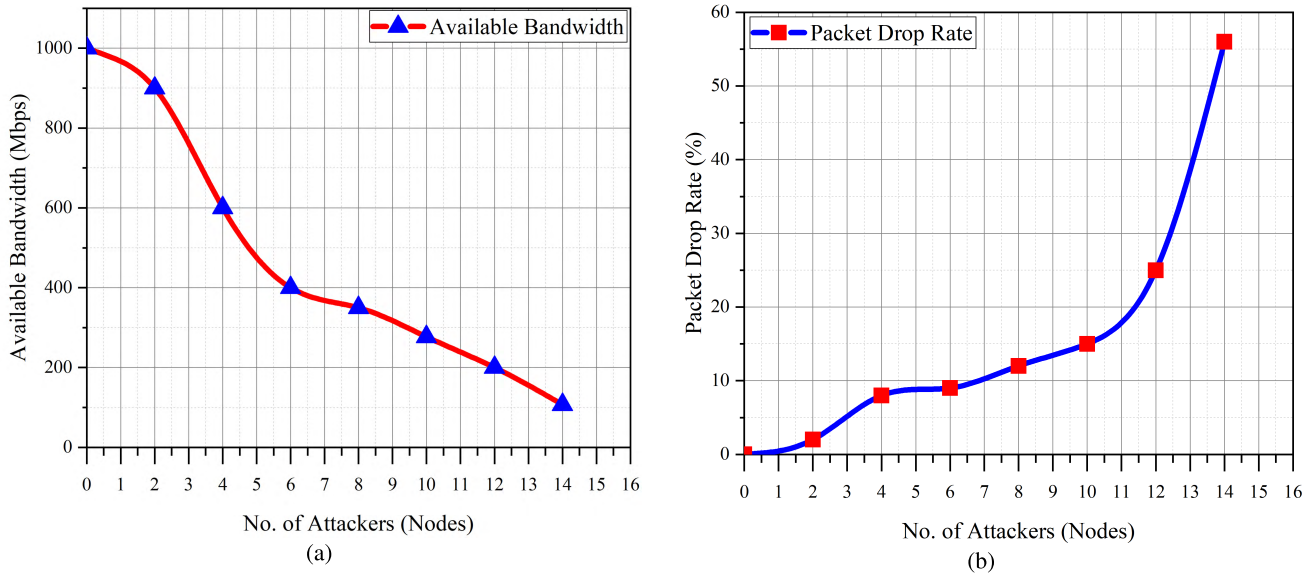

FIGURE 9. Effect of LFA on the link bandwidth saturation and packet drop rate.

network is increased to more than 40 percent, it indicates that the network performance has started to degrade. Similarly, an increase in link utilization ratio also indicates that the network is under LFA. We perform extensive experiments and evaluate CyberPulse using three different accuracy evaluation parameters. Initially, three machine learning performance evaluation metrics, i.e. *precision*, *recall*, and F_1 *score* were used for evaluation. Subsequently, the evaluation is performed using data partitioning, attribute selection, and using different classifiers.

1) CYBERPULSE EVALUATION USING PERFORMANCE EVALUATION METRICS

Three metrics for accuracy evaluation were used, i.e. *precision*, *recall*, and F_1 *score*. *Precision* is the measure of how close the predicted values are to the actual values. It is the value of the number of relevant flows retrieved divided by the total number of flows. The formula of accuracy is given in equation 7.

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

The results of accuracy evaluation metrics is given in Fig. 10. *Precision* values close to one are considered to be more accurate. It is pertinent to say that CyberPulse has been able to classify the traffic correctly into two categories. The accuracy of the two categories is more than 85%. *Recall* can be defined as the total number of relevant flows classified divided by the total number of retrieved flows. The formula of *recall* is given in equation 8.

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

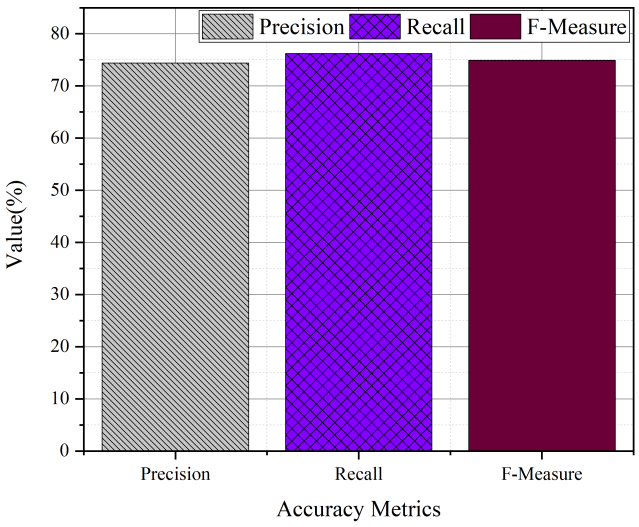

FIGURE 10. Evaluation of the ANN classifier using accuracy metrics.

CyberPulse performed slightly lower in the case of *recall* as compared to the *precision*. The *Flooding* flows were more accurately classified as compared to the *Legitimate* flows. The overall accuracy of the *recall* measure was around 95%. F_1 *score* is also an important evaluation measure as it combines both *precision* and *recall* values. The formula for F_1 *score* is given in equation 9.

$$F_1\ score = 2 \times \frac{Percision \times Recall}{Precision + Recall} \tag{9}$$

It can be observed from *precision* and *recall* in Fig. 10 that the F_1 *score* metrics performed slightly better as compared to *precision* and *recall*. The values of flood traffic detection are higher than that of *recall*. The overall accuracy of the F_1 *score* was around 76%. The accuracy of this metrics for legitimate traffic detection was approximately the same as the values of *precision* and *recall* metrics. Our results and analysis show that CyberPulse was able to accurately classify the traffic. Based on the classification values the Flood Mitigation module was able to eliminate the flood traffic. Overall it can be concluded that if the attacker is powerful and able to send high volumes of flood traffic than the available bandwidth of the system will be dropped as it can be observed from the Fig. 9a In the same way, if the flooding attack is increased, the packet drop rate will also tend to increase, severely affecting the legitimate traffic in the network. It can also be noted that CyberPulse effectively identifies the flows that are involved in the flooding of the network.

2) CYBERPULSE PERFORMANCE EVALUATION USING DATA PARTITIONING AND ATTRIBUTE SELECTION

Due to the novelty of our solution, we were unable to find relevant LFA classification techniques to directly compare with the proposed CyberPulse ML classification algorithm. In order to achieve a fair comparison of the results of our work we did however evaluate the accuracy of the system along several dimensions. We employed different: (1) state-of-the-art machine learning classification algorithms, (2) data partitioning strategies and (3) classification features and compared the results. Thereafter, we discuss the features of our solution with the most closely related works we could find.

In this section, we first evaluate the CyberPulse using different data splitting strategies and employing several attributes selection techniques. Initially, the data was split into multiple chunks and provided as input to the MLP algorithm to evaluate the effect of data partitioning strategies on the accuracy. The idea was to implement the best-suited strategy for the classification of CyberPulse flood traffic. We also perform evaluation by selecting critical attributes that play important role in the classification of the network traffic. We split data into three partitions with reference to training and testing i.e. 50% each, 70% training, and 90% training. Fig. 11 shows the evaluation results of data split, algorithm, and feature selection experiments. Similarly, it can be observed from the Fig. 11a that CyberPulse classifier performed well when the size of the training set was increased. There was a significant increase in the accuracy of the evaluation metrics when the training percentage of data was increased from 50% to 70%. The *precision* metrics rapidly

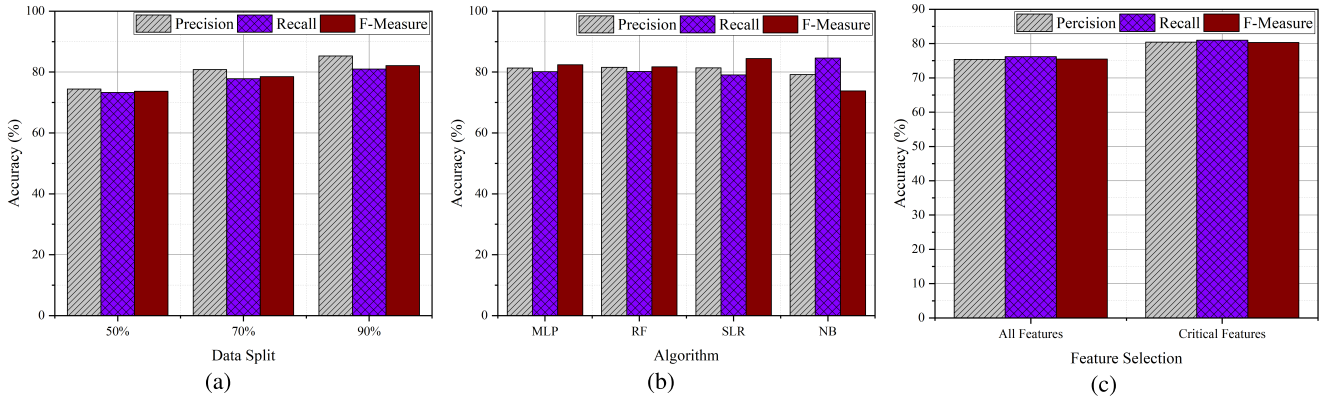

FIGURE 11. CyberPulse classifier evaluation using data split, critical features, and algorithm selection strategies.

increased to over 80% which was around 74% while 50% data was used for training. This comparison provides an insight for selecting a fair split of training and testing data to get better results. As it can be observed from the Fig. 11b that the correct selection of attributes plays a significant role in the accuracy of the classifier. We first performed the experiment by employing all the attributes and subsequently performed analysis by removing attributes such as node, the percentage of lost packet rate, lost bandwidth, packet size, and packet received. It can be clearly observed that the accuracy of the classifier was increased after removing those attributes.

3) CYBERPULSE PERFORMANCE EVALUATION USING DIFFERENT CLASSIFIERS

In this section, we provide a comparative analysis of our classification algorithm with respect to competing algorithmic approaches. In the MLP deep learning technique, there are multiple layers including, the input sensory layer, output layer, and one or more hidden layers that collaborate to extract salient features of the problem space. MLP has the ability to model and learn complex non-linear relationships of the given domain. Therefore, it was best suited in our case where some attributes of the network traffic were not linearly dependent on each other such as maximum bandwidth and packet drop rate. To validate the classifier, the comparison was performed to analyze the validity of the results using the MLP classifier with three different classification algorithms i.e. Random Forest (RF), Simple Logistics Regression (SLR), and Naïve Bayes (NB). The reason for using the NB algorithm for comparison was that it has been used as a baseline method for several classification techniques in the past due to its simplicity and ease of implementation [12].

SLR and RF have also been widely used for classification of real-time data because of their good predictive performance and excellent comprehensibility [32]. It can be observed from Fig. 11c that the values of *precision*, *recall*, and F_1 *score* are approximately the same for all the algorithms. We observed that the value of *recall* changed in case of SLR and NB, where it dropped and increased respectively in both algorithms. However, there is a big difference in other performance metrics between the SLR, and NB algorithms. It is also noted that the value of F_1 *score* dropped in the NB algorithm because the NB classifier considers all the attributes to be independent and there is a very minor impact on the value of accuracy when the attributes are dependent on each other [33]. While NB provides the best *Recall*, overall MLP performed better for all the accuracy metrics. As we were interested in the classification of network traffic, therefore the overall accuracy of the classification was of foremost importance. Therefore considering our requirements, MLP algorithm was the best suited as it performed well and provided reasonably good cumulative results.

C. EFFECT ON LFA MITIGATION USING CYBERPULSE

After successful classification of the flood traffic, the responsible flows are identified. This information is sent to the Flood Mitigation module which terminates the flows using null routing technique. The flows are dropped and not forwarded to any further route by configuring the null route with a route flag. Null routing technique was chosen because it is a simplified technique and is available on all network routers with no performance impact on the network.

VII. RELATED WORK

LFA is a dangerous flooding attack that has the ability to congest SDN interfaces and links connecting to other layers. It has also gained a lot of attention during recent years [34]. In SDN, the traffic path is decided by the controller where the packets are forwarded according to the flow rules installed on the switches. Hence, traditional LFA mitigation techniques become invalid in such situations. CyberPulse intends to provide defense against control channel LFAs. This section discusses the existing LFA mitigation techniques and categorises as follows:

A. LFA MITIGATION USING LINK PROBING TECHNIQUES

Link probing techniques employ surveillance of the target network links to probe for malicious flows. Wang et al.

propose a link obfuscation mechanism applied at the time of link creation. LinkScope has been proposed in [9] by Xue *et al.* which inspects links that contain flows involved in network congestion using hop by hop and end to end network measurement. SPIFFY [11] temporarily increases network bandwidth and measures the network flows before and after the bandwidth expansion phase. Only legitimate users adapt to the bandwidth change, however, the adversaries end up consuming all their bandwidth during the expansion phase allowing them to be easily detected. A downside to this approach is that legitimate traffic that is unable to adapt to the bandwidth expansion and compression step, will end up being treated as malicious. The SDN HoneyNet technique [13] calculates the betweenness centrality of the links in the network to identify the ones that can become a bottleneck. Subsequently, it deploys a HoneyNet topology to provide a fake link map to the adversaries. CoDef is based on the collaboration among different Autonomous Systems (AS) in the network where the AS that are not under attack handle the legitimate traffic from attacked AS by creating a bypass link around the attacked location [14].

B. LFA MITIGATION USING TRAFFIC ENGINEERING PRINCIPLES

Traffic engineering is a popular mechanism for mitigating a wide variety of network attacks. Takayuki *et al.* [16] propose increase in traceroute packets volume to detect LFA. However, it is complex to distinguish between legitimate and malicious traceroute commands in the network. Liaskos *et al.* [15] propose a traffic engineering mechanism which uses relational algebra principles for LFA mitigation. When the attack occurs the defender reroutes the traffic to avoid congestions. The defender keeps on repeating this process until the attacker's identity is exposed by knowing the sources that are participating in network flooding multiple times. Gillani *et al.* propose dynamic network resource allocation using virtual networks placement during LFA [35]. Network resources are constantly migrated to alleviate LFA. Kalliola *et al.* propose legitimate traffic features learning, elastic capacity invocation, and blacklisting malicious hosts to provide defense during LFA [36].

C. LFA MITIGATION USING SDN PRINCIPLES-BASED APPROACHES

Some researchers utilize the benefit of having a centralized controller with the ability to observe all the switches it controls and their corresponding flows to avoid LFA in SDN. A flow table inspection technique has been proposed by Xiao *et al.* [37]. Flow table inspection is performed in order to identify malicious flows where the bloom filtering technique is utilized by a detector module to detect malicious adversaries. Adversaries create a link map of the network to select a specific link. It is difficult for an adversary to locate target link before the attack if the link information keeps on changing. Wang *et al.* [19] propose Woodpecker which employs incremental SDN deployment to mitigate LFA. A network probing technique is employed to locate LFA and subsequently, routers at that location are upgraded to SDN switches to increase network connectivity. Finally, by employing SDN principles of centralized network management, the traffic is balanced in the network. SDN-based traffic maneuvering technique has been proposed by Aydeger *et al.* [20]. Link obfuscation is performed when a threat has been identified to make it difficult for the adversary to create a correct link map of the network. However, link obfuscation causes legitimate traffic delays because new paths may not always be optimal.

Recently, an efficient DoS mitigation technique called FloodDefender has been proposed by [38]. It eliminates DoS attacks by table miss analysis, flow rule inspection, and packet filtering techniques. However, such mechanisms render invalid in case of LFA which uses low rate legitimate traffic. In the same way, FloodDefender transfers table-miss packets to neighbor switches in order to protect the communication link from being jammed. However, our approach is based on ML whereby the controller employs the ML strategy to decide whether the flows are malicious or not and takes subsequent action.

In summary, most of the link probing techniques discussed above assume traditional networks. Researcher who have performed experiments on SDN have merely used SDN testbeds or SDN-based techniques to defend against LFAs only. From the best of our knowledge there is no prior technique that specifically addresses LFA on SDN backbone networks context. We have identified this weakness in the SDN architecture against LFA and proposed a solution that can effectively alleviate LFA on control channel of SDN. Therefore, we present CyberPulse a novel ML-based LFA mitigation technique which employs link inspection mechanism to detect link congestion caused by LFA. CyberPulse will help in providing efficient network infrastructure security and enable virtualization.

The accuracy of our experimental results in terms of evaluation metrics is slightly lower because of the limitation of the training dataset, it is a challenge to get dataset specially managed for LFA. So, we modified the dataset for BHP flooding attacks [28] and utilized it for LFA classification. We are performing extensive experiments to address this drawback in order to increase the evaluation accuracy, and the outcome will be published in the future work. This paper provides the basis to design and architecture of a solution that collects network statistics, performs training, and demonstrates potential to alleviate flood traffic. In this regard, we are further developing an extended ML-based solution against LFA on SDN which will work at line speed and perform LFA defense and mitigation in real-time.

VIII. CONCLUSION

SDN continues to grow in popularity for the complex datacenter, enterprise and more recently WAN environments. The provision of separate data, control, and application planes as well a control channel allows flexibility in management and application development for complex networks.

However, this very strength also doubles as the single greatest security weakness for this new paradigm in networking. Link flooding attacks that have previously been identified as dangerous for traditional networks become crippling if used against the control channel in SDN. LFA control channel attacks can be devastating because they are stealthy, are relatively easy to conduct, and have the ability to completely cripple the network. Moreover, they are very hard to detect and mitigate by traditional means. The novel contribution of this work is that it utilizes the power of machine and deep learning and its ability to empower machines to automatically learn about the telltale statistical features of an LFA attack and then automatically identify the same in a network big data during an actual attack.

In this paper, we have motivated the need to address SDN-based LFA with the help of experiments that highlight the impact of a successful attack. Furthermore, we proposed CyberPulse, a lightweight SDN controller extension for securing the SDN control channel against LFA that utilizes machine learning and deep learning techniques. An early prototype of CyberPulse is implemented on the application plane to monitor the control channel for keeping track of ongoing traffic flows. For each flow in the network, CyberPulse examines the statistics on the control channel and classifies the network traffic using a deep learning multi-layer perception technique. Finally, it is demonstrated to be able to effectively drop the classified flows and facilitate the seamless operation of the network using null routing. One of the main advantages of CyberPulse is that it does not sacrifice the legitimate flows in the network. Based on the evaluation it can be concluded that CyberPulse accurately identifies traffic flows that exhibit LFA characteristics and mitigates the attack efficiently. A limitation, however, is the added complexity of employing an ML approach. Nevertheless, the cost-benefit trade-off is worth using this technique to safeguard current Software-Defined internet infrastructure from LFAs. An extension of the proposed framework is underway, and the detailed design and evaluation will be presented in future work.

IX. ACKNOWLEDGMENT

This work was supported by the Deanship of Scientific Research (DSR) at King Faisal University under Grant 186147.

REFERENCES

[1] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Commun. Surv. Tuts.*, vol. 16, no. 4, pp. 2181–2206, 4th Quart., 2014.

[2] B. Görkemli, A. M. Parlakışık, S. Civanlar, A. Ulaş, and A. M. Tekalp, "Dynamic management of control plane performance in software-defined networks," in *Proc. IEEE Int. NetSoft Conf. Workshops (NetSoft)*, Jun. 2016, pp. 68–72.

[3] R. Mohammadi and R. Javidan, "An adaptive type-2 fuzzy traffic engineering method for video surveillance systems over software defined networks," *Multimedia Tools Appl.*, vol. 76, no. 22, pp. 23627–23642, 2017.

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

[5] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2013, pp. 2211–2219.

[6] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[7] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 623–654, 1st Quart., 2016.

[8] L. Wei and C. Fung, "Flowranger: A request prioritizing algorithm for controller DoS attacks in software defined networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2015, pp. 5254–5259.

[9] L. Xue, X. Ma, X. Luo, E. W. W. Chan, T. T. Miu, and G. Gu, "LinkScope: Toward detecting target link flooding attacks," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 10, pp. 2423–2438, Oct. 2018.

[10] X. Ma, J. Li, Y. Tang, B. An, and X. Guan, "Protecting internet infrastructure against link flooding attacks: A techno-economic perspective," *Inf. Sci.*, vol. 479, pp. 486–502, Apr. 2018.

[11] M. S. Kang, V. D. Gligor, and V. Sekar, "SPIFFY: Inducing cost-detectability tradeoffs for persistent link-flooding attacks," in *Proc. NDSS*, 2016, pp. 1–15.

[12] X. Chen, G. Zeng, Q. Zhang, L. Chen, and Z. Wang, "Classification of medical consultation text using mobile agent system based on Naïve Bayes classifier," in *Proc. Int. Conf. 5G Future Wireless Netw.* Beijing, China: Springer, 2017, pp. 371–384.

[13] J. Kim and S. Shin, "Software-defined HoneyNet: Towards mitigating link flooding attacks," in *Proc. 47th IEEE/IFP Int. Conf. Dependable Syst. Netw. Workshop (DSN-W)*, Jun. 2017, pp. 99–100.

[14] S. B. Lee, M. S. Kang, and V. D. Gligor, "CoDef: Collaborative defense against large-scale link-flooding attacks," in *Proc. 9th ACM Conf. Emerg. Netw. Exp. Technol.*, 2013, pp. 417–428.

[15] C. Liaskos, V. Kotronis, and X. Dimitropoulos, "A novel framework for modeling and mitigating distributed link flooding attacks," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[16] T. Hirayama, K. Toyoda, and I. Sasase, "Fast target link flooding attack detection scheme by analyzing traceroute packets flow," in *Proc. IEEE Int. Workshop Inf. Forensics Security (WIFS)*, Nov. 2015, pp. 1–6.

[17] L. Xue, X. Luo, E. W. Chan, and X. Zhan, "Towards detecting target link flooding attack," in *Proc. 28th Large Installation Syst. Admin. Conf. (LISA)*, 2014, pp. 90–105.

[18] D. Gkounis, V. Kotronis, C. Liaskos, and X. Dimitropoulos, "On the interplay of link-flooding attacks and traffic engineering," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 1, pp. 5–11, 2016.

[19] L. Wang, Q. Li, Y. Jiang, and J. Wu, "Towards mitigating link flooding attack via incremental SDN deployment," in *Proc. IEEE Int. Symp. Comput. Commun. (ISCC)*, Jun. 2016, pp. 397–402.

[20] A. Aydeger, N. Saputro, K. Akkaya, and M. Rahman, "Mitigating crossfire attacks using SDN-based moving target defense," in *Proc. 41st IEEE Int. Conf. Local Comput. Netw. (LCN)*, Nov. 2016, pp. 627–630.

[21] M-Team. (2018). *A Virtual Network on Your Desktop*. [Online]. Available: http://mininet.org/

[22] (2018). *Project Floodlight*. [Online]. Available: http://www.projectfloodlight.org/floodlight/

[23] R. Mohammadi, R. Javidan, and M. Conti, "SLICOTS: An SDN-based lightweight countermeasure for TCP SYN flooding attacks," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 2, pp. 487–497, Jun. 2017.

[24] K. Bu, X. Wen, B. Yang, Y. Chen, L. E. Li, and X. Chen, "Is every flow on the right track?: Inspect SDN forwarding with rulescope," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[25] X. Wen *et al.*, "SDNshield: Reconciliating configurable application permissions for SDN app markets," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun./Jul. 2016, pp. 121–132.

[26] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: New attacks and countermeasures," in *Proc. NDSS*, vol. 15, 2015, pp. 8–11.

[27] O. N. Foundation. (2014). *OpenFlow Specification*. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf

[28] D. Dheeru and E. K. Taniskidou. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[29] NCT University. (2018). *Estinet Network Emulator*. [Online]. Available: http://nsl.cs.nctu.edu.tw/NSL/nctuns.html
[30] HP Company. (1999). *Utilization, HP Toptools for Hubs & Switches*. [Online]. Available: http://hp.com/rnd.htm
[31] SharkFest. (2018). *Wireshark*. [Online]. Available: https://www.wireshark.org/
[32] A. De Caigny, K. Coussement, and K. W. De Bock, "A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees," *Eur. J. Oper. Res.*, vol. 269, pp. 760–772, Sep. 2018.
[33] J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the poor assumptions of naive Bayes text classifiers," in *Proc. ACM Int. Conf. Assoc. Adv. Artif. Intell.* Washington, DC, USA: AAI Press, 2003, pp. 616–623.
[34] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS attack detection and mitigation using SDN: Methods, practices, and solutions," *Arabian J. Sci. Eng.*, vol. 42, no. 2, pp. 425–441, 2017.
[35] F. Gillani, E. Al-Shaer, S. Lo, Q. Duan, M. Ammar, and E. Zegura, "Agile virtualized infrastructure to proactively defend against cyber attacks," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr./May 2015, pp. 729–737.
[36] A. Kalliola, K. Lee, H. Lee, and T. Aura, "Flooding DDoS mitigation and traffic management with software defined networking," in *Proc. 4th IEEE Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2015, pp. 248–254.
[37] P. Xiao, Z. Li, H. Qi, W. Qu, and H. Yu, "An efficient DDoS detection with bloom filter in SDN," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2016, pp. 1–6.
[38] G. Shang, P. Zhe, X. Bin, H. Aiqun, and R. Kui, "FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

RAIHAN UR RASOOL is currently a Fulbright Alumnus with the University of Chicago, USA. He is also with Victoria University, Melbourne. His research interests include large-scale systems, security and computer architecture. His research work comprising over 60 articles published in various international conferences and journals.

USMAN ASHRAF received the B.S. degree in computer science from FAST Lahore, in 2003, and the M.S. and Ph.D. degrees in computer networks from INSA Toulouse, France, in 2006 and 2010, respectively. He is currently with the College of Computer Science and IT, King Faisal University. He has several publications in prestigious international journals, including the IEEE Communications Letters and the IEEE Transactions on Mobile Computing. He has more than seven years of teaching and research experience. In his last position, he chaired the Department of Computer Science, Air University Islamabad.

KHANDAKAR AHMED received the Ph.D. degree from RMIT University, Australia, in 2014. During his Ph.D., he was an Active Scholar with the Network Research Group, School of Engineering, where he explored data-centric storage in wireless sensor networks. He is currently a Lecturer with the Discipline of IT, School of Engineering and Science, Victoria University, Melbourne, Australia. Before joining Victoria University, he was a Full-Time Lecturer with the School of IT and Engineering (SITE), Melbourne Institute of Technology (MIT), Melbourne, and a Researcher in Australia—India Research Centre for Automation Software Engineering (AICAUSE), RMIT University, from 2016 to 2017. He has been with AICAUSE as a Postdoctoral Researcher before he takes the full-time position with MIT, from 2015 to 2016. He is also serving as a member of the Editorial Board for the *Australian Journal of Telecommunications and the Digital Economy* and TPC chair for ITNAC 2017. He has also been serving as a Reviewer for several international A star journal's and conferences.

HUA WANG received the Ph.D. degree in computer science from the University of Southern Queensland, in 2004, where he was a Professor, from 2011 to 2013. He is currently a Full-Time Professor with the Centre for Applied Informatics, Victoria University. He has authored or co-authored over 150 peer-reviewed research papers mainly in data security, data mining, access control, privacy and Web services, as well as their applications in the fields of e-health and e-environment.

WAJID RAFIQUE received the B.S. degree in computer sciences degree from Virtual University, Pakistan, and the M.S. degree in software engineering from the National University of Sciences and Technology (NUST), Pakistan. He is currently pursuing the Ph.D. degree in computer sciences with the Department of Computer Science and Technology, Nanjing University, China. His research interests include big data services, machine learning, mobile cloud computing, and network security.

ZAHID ANWAR received the M.S. and Ph.D. degrees in computer sciences from the University of Illinois at Urbana-Champaign, in 2005 and 2008, respectively. He was a Software Engineer and Researcher with IBM, Intel, Motorola, National Center for Supercomputing Applications (NCSA), xFlow Research, and CERN on various projects related to information security, operating systems design, and data analytics. He holds Postdoctorate experience from Concordia University. In the past, he was a Faculty Member with the University of North Carolina at Charlotte. He is currently a Faculty Member with the National University of Sciences and Technology and an Associate Professor with Fontbonne University.