# CyberPulse: A Security Framework for Software-Defined Networks

Thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

Institute for Sustainable Industries and Liveable Cities (ISILC)

VU Research

Victoria University, Melbourne

By

Raihan Ur Rasool

March 2020

## Abstract

Software-Defined Networking (SDN) technology provides a new perspective in traditional network management by separating infrastructure plane from the control plane which facilitates a higher level of programmability and management. While centralized control provides lucrative benefits, the control channel becomes a bottleneck and home to numerous attacks. We conduct a detailed study and find that crossfire Link Flooding Attacks (LFA) are one of the most lethal attacks for SDN due to the utilization of low-rate traffic and persistent attacking nature. LFAs can be launched by the malicious adversaries to congest the control plane with low-rate traffic which can obstruct the flow rule installation and can ultimately bring down the whole network. Similarly, the adversary can employ bots to generate low-rate traffic to congest the control channel, and ultimately bring down the control plane and data plane connection causing service disruption.

We present a systematic and comparative study on the vulnerabilities of LFAs on all the SDN planes, elaborate in detail the LFA types, techniques, and their behavior in all the variant of SDN. We then illustrate the importance of a defense mechanism employing a distributed strategy against LFAs and propose a Machine Learning (ML) based framework namely CyberPulse. Its detailed design, components, and their interaction, working principles, implementation, and in-depth evaluation are presented subsequently.

This research presents a novel approach to write anomaly patterns and makes a significant contribution by developing a pattern-matching engine as the first line of defense against known attacks at a line-speed. The second important contribution is the effective detection and mitigation of LFAs in SDN through deep learning techniques. We perform twofold experiments to classify and mitigate LFAs. In the initial experimental setup, we utilize Artificial Neural Networks backward propagation technique to effectively classify the incoming traffic. In the second set of experiments, we employ a holistic approach in which CyberPulse demonstrates algorithm agnostic behavior and employs a pre-trained ML repository for precise classification. As an important scientific contribution, CyberPulse framework has been developed ground up using modern software engineering principles and hence provides very limited bandwidth and computational overhead. It has several useful features such as large-scale network-level monitoring, real-time network status information, and support for a wide variety of ML algorithms. An extensive evaluation is performed using Floodlight open-source controller which shows that CyberPulse offers limited bandwidth and computational overhead and proactively detect and defend against LFA in real-time.

This thesis contributes to the state-of-the-art by presenting a novel framework for the defense, detection, and mitigation of LFA in SDN by utilizing ML-based classification techniques. Existing solutions in the area mandate complex hardware for detection and defense, but our presented solution offers a unique advantage in the sense that it operates on real-time traffic scenario as well as it utilizes multiple ML classification algorithms for LFA traffic classification without necessitating complex and expensive hardware. In the future, we plan to implement it on a large testbed and extend it by training on multiple datasets for multiple types of attacks.

## DOCTOR OF PHILOSOPHY DECLARATION

I, Raihan ur Rasool, declare that the Ph.D. thesis entitled *CyberPulse: A Security Framework for Software-Defined Networks* is no more than 100, 000 words in length including quotes and exclusive of tables, figures, appendices, bibliography, references and footnotes.

This exegesis contains no material that has been submitted previously, in whole or in part, for the award of any other academic degree or diploma. Except where otherwise indicated, this thesis is my own work.

Signature [redacted] Date 01/03/2020

## ACKNOWLEDGEMENTS

This work would not have been possible without the support of the Institute for Sustainable Industries and Liveable Cities (ISILC) at Victoria University (VU). I am especially indebted to Dr. Hua Wang and Dr. Khandakar Ahmed who have been supportive of my career goals. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better advisors and mentors. I am grateful to Sudha Subramani and Sarath Kumar for being there for me in need and for the fun-time we spent together.

Nobody has been more important to me in the pursuit of this research than the members of my family. I would like to thank my parents; whose love and guidance are with me in whatever I pursue. They are the ultimate role models. Most importantly, I wish to thank my loving and supportive wife, Rabia Saleem, and my two children, Amna and Ayna. This journey would not have been possible if not for them, and I dedicate this milestone to them.

## PUBLICATIONS

### Journal Articles

*(In Review)*

1. **Raihan Ur Rasool**, Hua Wang, Usman Ashraf, Khandakar Ahmed, Zahid Anwar, Wajid Rafique, “A Survey of Link Flooding Attacks in Software Defined Network Ecosystems”. Journal of Network and Computer Applications, 2020.
2. **Raihan Ur Rasool**, Khandakar Ahmed, Zahid Anwar, Hua Wang, Usman Ashraf, “CyberPulse++: A Machine Learning-Based Security Framework for Detecting Link Flooding Attacks in Software Defined Networks”. IEEE Transactions on Cybernetics, 2019.

*(Published)*

1. Wajid Rafiq, **Raihan Ur Rasool**, Wanchun Dou, "Using Edge Computing: Requirements Standardization, and Security Challenges", IEEE Communications Surveys and Tutorials, 2020.
2. **Raihan Ur Rasool**, Usman Ashraf, Khandakar Ahmed, Hua Wang, Wajid Rafique, Zahid Anwar, “CyberPulse: A Machine Learning-Based Link Flooding Attack Mitigation System for Software-Defined Networks”. IEEE Access, vol. 7, pp. 34885-34899, 2019.
3. **Raihan Ur Rasool**, Maleeha Najam, Hafiz Farooq Ahmad, Hua Wang, Zahid Anwar, “A novel JSON based regular expression language for pattern matching in the internet of things”. The Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 4, pp. 1463–1481, 2018.
4. Abdul Majeed, **Raihan ur Rasool**, Farooq Ahmad, Masoom Alam, Nadeem Javaid, “Near-miss situation based visual analysis of SIEM rules for real-time network security monitoring”, Journal of Ambient Intelligence & Humanized Computing, 10: 1509, 2018.
5. Zahra Ali, **Raihan ur Rasool**, Peter Bloodsworth, Shamyl Bin Mansoor, “Facebook-based cloud resource sharing”, Computers and Electrical Engineering, vol 66, pp 162-173, 2018.

6. Asad W. Malik, **Raihan ur Rasool**, Zahid Anwar, Shahid Nawaz, “A Generic Framework for Application Streaming Service”, Computers & Electrical Engineering, vol 66, pp 149-161, 2018.
7. Sarah Shafqat, Saira Kishwer, **Raihan Ur Rasool**, Junaid Qadir, “Big data analytics enhanced healthcare systems: a review”, The Journal of Supercomputing, pp 1-46, 2018.
8. Amna Riaz, Junaid Qadir, Usman Younis, **Raihan Ur Rasool**, Hafiz Farooq Ahmad, Adnan K. Kiani, “Intrusion Detection Systems in Cloud Computing: A Contemporary Review of Techniques and Solutions”. Journal of Information Science and Engineering. 33(3): 611-634, 2017.
9. Syeda ZarAfshan Gohera, Peter Bloodsworth, **Raihan Ur Rasool**, Richard McClatchey, "Cloud provider capacity augmentation through automated resource bartering", Future Generation Computer Systems, vol. 81, pp 203-218, 2017.
10. Zia-Ul-Qamar, **Raihan Ur Rasool**, Hammad Majeed, "Probability-based measure to calculate reliability of multi-layer web applications", International Journal of Internet Protocol Technology, pp 224-235, 2017.
11. Anwaar Ali, Junaid Qadir, **Raihan ur Rasool**, Jon Crowcroft, "Big Data For Development: Applications and Techniques", Journal of Big Data, vol 1, 2016.
12. Jon Crowcroft Junaid Qadir, Anwaar Ali, **Raihan ur Rasool**, Andrej Zwitter, Arjuna Sathiaseelan, "Crisis Analytics: Big Data Driven Crisis Response", Springer-Open Journal of International Humanitarian Action, 2016.
13. Maleeha Najam, **Raihan Ur Rasool**, Hafiz Farooq Ahmad, Usman Ashraf, Asad Waqar Malik,"Pattern Matching for DNA Sequencing Data Using Multiple Bloom Filters", BioMed Research International Journal, 2019

**Published Conference Articles**

1. **Raihan Ur Rasool**, Hua Wang, Wajid Rafique, Jianming Yong, Jinli Cao, “A Study on Securing Software Defined Networks”. In Proc. Intl. Conf. on Web Information Systems Engineering (WISE), Moscow Russia, pp. 479-489, 2017.
2. Fatimah Abdualaziz, Noor Zaman, **Raihan Rasool**, "Energy Efficient Middleware: Design and Development for Mobile Applications", In Proc. 19th International Conference on Advanced Communication Technology (ICACT), 2017.

**Table of Contents**

**List of Figures**

## List of Tables

# CHAPTER 1. INTRODUCTION

Software-Defined Networking (SDN) provides an open interface for the development of the software to control the network infrastructure. It enables flexible control of traffic using the programmability, moreover, it leverages the softwarization characteristics to modify, inspect, and manage the traffic at runtime. All these functions are provided in terms of abstract services at different layers of SDN. Link Flooding Attacks (LFAs) have emerged as one of the stealthiest attacks on modern networks. These attacks can cause a Denial of Service (DoS) by choking important links and ultimately bringing the whole network down [1]. Depending on the technique and methodology, several LFAs have broadly been described in the literature [2] such as crossfire [3], coremelt [4], and spamhaus attacks [5]. Among these, a crossfire LFA is harder to detect as it isolates the target by flooding the links around it with low-rate legitimate traffic. Such attacks have the potential of disrupting the most widely used SDN in a variety of ways [3].  Current sensor networks operate on constant sensor measurements of the underlying physical infrastructure where they are attached like temperature sensors, humidity operators, lights, fluid level indicators, and many more.

A huge amount of data is produced by these sensors which need to be transferred to remote locations for efficient processing and information extraction. This data, trigger events which subsequently relate to certain actions. Data is also filtered at certain locations for security and privacy purposes which operates on pattern matching techniques. Therefore, pattern matching is essential in the realization of Ambient Intelligence (Aml) and humanized computing. A lightweight pattern matching solution provides the first line of defense against the lethal security attacks. Pattern matching works in a way that it searches for a specific pattern of traffic in the network. Modern network security solutions for examples Network Intrusion Detection System (NIDS), firewall and Intrusion Detection System (IDS) routinely deploy Deep Content Inspection (DCI) techniques to ascertain malicious traffic.  In this chapter, we discuss an overview of SDN, LFA, pattern matching technology, and their variants in the networking paradigm. Similarly, we explain the vulnerabilities of LFA in SDN, particularly, we focus on control channel LFA as how they are lethal for modern network deployments. We provide instances of different attacks on SDN layers moreover, the attack vulnerabilities on different SDN interfaces are discussed.

In the same way, at the end of this chapter, the contributions of this research are formally presented.

## 1.1 SDN Architecture

The primary architecture of the SDN comprises up of infrastructure, control, and application layers. The infrastructure layer comprises up of hardware elements of the network including switches, routers, and gateways which expose their connectivity with the controller using the southbound interfaces often called as Control-Data Plane Interface (C-DPI). The network requirements are implemented at the application layer and are exposed to the controller using the northbound interface (NBI).

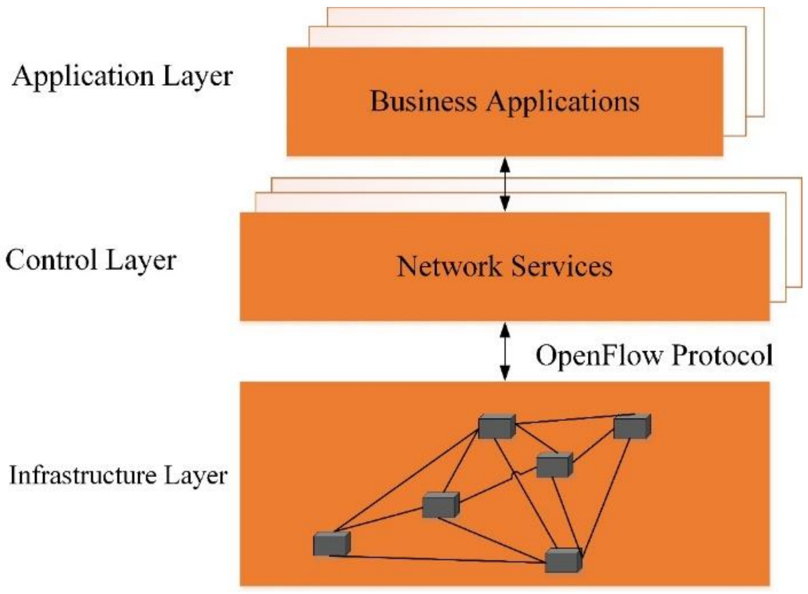


Figure 1-1 SDN architecture, the figure at [70] has been redrawn and expanded.

The controller accepts the high-level network requirements from the application layer and translates them into low-level commands for the network elements. The infrastructure layer operates on the commands sent by the controller and forwards the network traffic. The infrastructure layer works barely as the forwarding element as all the intelligence has been shifted at the controller of the SDN. However, a limited subset of control and management functions have also been provided for localized control. The SDN controller manages the network according to application-level policy requirements.

Figure 1-1 illustrates the high-level architecture of SDN representing the three planes. SDN calls for the separation of the concerns where the management decisions have been implemented at the application layer of the SDN. In contrast to the traditional networks where the forwarding devices have the control and data plane functionality at the infrastructure, the centralized management offers a broader perspective of the network which eases the decision-making process and provides granular control. The abstract network services provide programmability of the network where the application plane instigate the required services to the controller which delivers them according to the global view of the network.

### *1.1.1 Data Plane*

The resources in data plane deal with the customer traffic as well as the auxiliary resources to enable Quality of Service (QoS) requirements. The data plane normally contains the network elements which comprise up of traffic source and sinks, moreover it contains the virtualizer which abstracts the data plane services and provides an interface to the controller to enforce network connectivity and policy. The traffic in the network can enter or leave on physical or logical ports and can be forwarded to the processing units. The decision making in the data plane is not autonomous where it is managed by the controller, however, the controller can facilitate the data plane to respond autonomously to the events such as network failure, policy consistency, or by proactive flow rule installation. The C-DPI performs the function of the network capabilities advertisement, event notification, and programmatic control of the data plane elements. The data plane agents are the key contributors to the implementation of the controller instructions. Moreover, the data plane coordinator performs the data plane resource allocation to the client agents and ensure policy to use them. Agents and coordinators exist on all the planes of the SDN and serve the same functionality. At a low-level, the data plane elements are the hardware elements whereas at the highest level of abstraction they are the software services exposed by the layer because SDN operates on an abstraction model. Figure 1-2 represents a detailed SDN architecture illustrating the OpenFlow (OF) switches at the data plane of SDN.

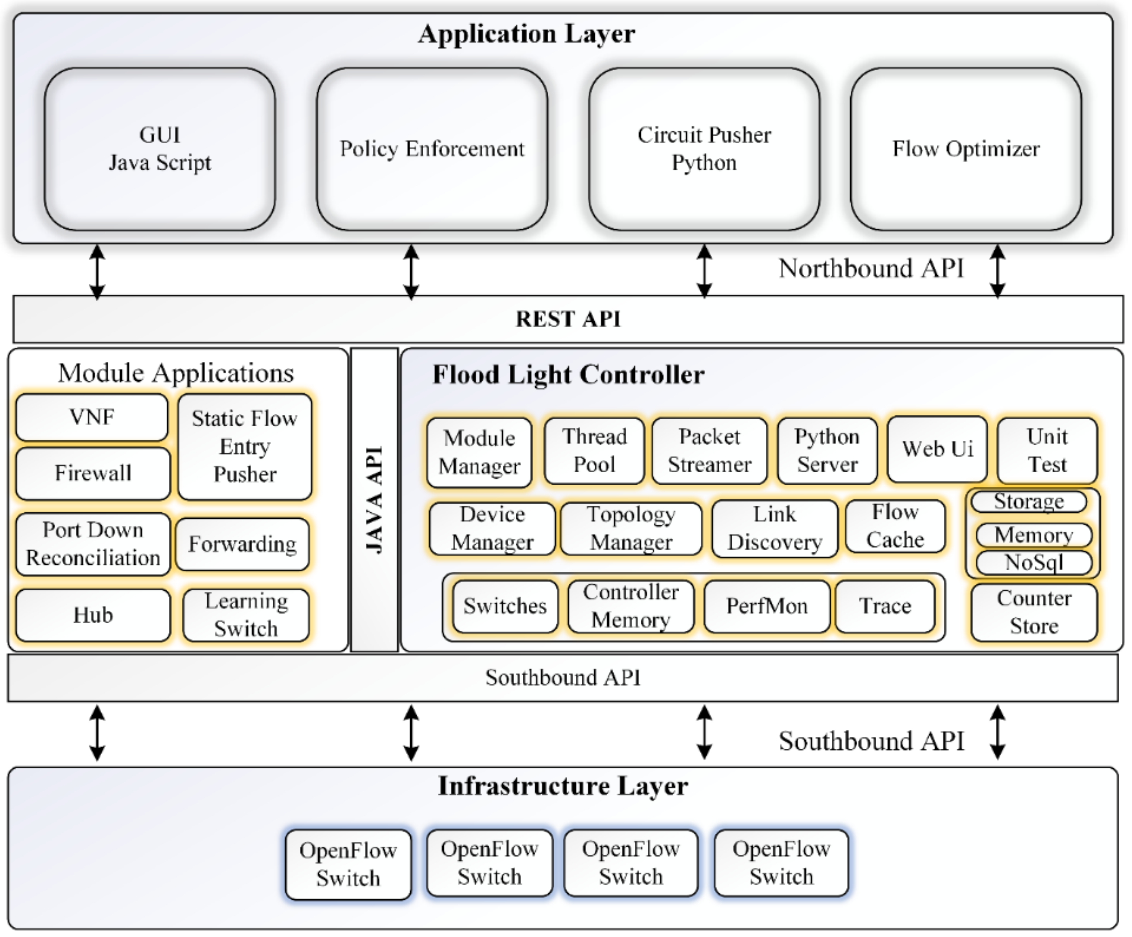


Figure 1-2 The reference architecture of SDN.

### *1.1.2 Control Plane*

The SDN control is logically centralized where the controller usually operates on the subnet scope of the network which spans on more than single network element. The services and the functions of the network comprise up of controller’s externally observable behavior. Other functionalities depend on the circumstances e.g., topology knowledge and optimal path computation. In the distributed control of the network resources, the control messages portray a high amount of overhead on the network. The control consists of infrastructure plane control, virtualizer, coordinators, and agents. The data plane control function known as orchestrator provides an abstract singular controller view to the multiple network elements. The coordinator acts as manager which setup the client-server environment. The virtualizer allocates abstract resources to particular clients and the applications. The controller agent model acts in a controller and controlled entity. The agent is supervised by the control component of the controller which illustrates the resources of the client from the server's perspective.

### *1.1.3 Application Plane*

Applications in SDN allows the behavior and resources to specify the required network functionality. The applications in the application plane can invoke other services and has the ability to utilize other controllers to accomplish the objectives. The interface between the application and the data plane is called the Application-Control Plane Interface (A-CPI). The applications in the application plane also support an A-CPI agent which enables for a recursive application hierarchy.  The latitude of hierarchies depends on the level of abstraction on which they exist. The application plane typically sends commands to alter the network state, and queries the network state. The A-CPI can also be used for additional functions of an input to control virtualized network functions. SDN provides an open interaction paradigm where software development is performed to flexibly control the communication in a network resource provision and control of network traffic. Moreover, it enables efficient surveillance in the network and real-time modification to efficiently orchestrate the services and provide security which is one of the prime concern in the current network paradigm.

## 1.2 Security in SDN

SDN was proposed to deal with the increasing network complexities and size due to the continuous development in current networks. SDN reduces network complexity by providing simplified and centralized network management and offers separate layers for data and control planes that help in flexible and dynamic network operation [6-8]. In SDN, data plane can only provide the traffic forwarding functionality while centralized controller has the global view of the whole network which can be programmed for desired traffic forwarding [9], [10], [11]. SDN utilizes OF protocol for communication between the SDN layers [12]. Every OF-enabled device in the data plane has flow tables that are managed by the controller. Flow tables contain the entries called flow rules to route the traffic to its destined path. All the incoming packets are compared with the flow entries in the flow table if flow entry for a flow is not found, it forwards the packet to the controller querying about the further action. The controller then updates the rules after the packet inspection in the flow table 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 leads to serious issues of security. A crafted adversary can cause flooding by generating a large number of different header fields [13] [14], [15]. A more serious threat is when a control channel is attacked creating congestion on the link with anomalous traffic using LFA [16].

LFA is one of the newest forms of flooding attacks which disrupt communication on the underlying link. As the name implies that it is a link-based attack, where the link connecting to the target server is flooded to cause traffic congestion which ultimately disrupts the legitimate traffic to the target server. Initially, the attacker identifies the target server and creates a link map of the target by sending traceroute commands [17]. Subsequently, specific servers are carefully selected that lie around the path of the target link, these servers are called, decoy servers. After the identification of the decoy servers, a number of bots are selected which can generate sufficient traffic to flood the link. Finally, the decoy servers are manipulated by bots to send low-rate traffic to each other in order to cause the congestion on the target link which will ultimately disrupt the communication of the target server with the rest of the network.

Recently, some studies have been performed to mitigate LFA in SDN. A few of the available studies focus on providing fake network map to the adversaries [2], [18], [19]. These techniques operate by exposing fake network topology to the attackers. They utilize graph metric techniques to identify the critical links that are vulnerable to being attacked and afterward present fake links to the attacker making the attack ineffective. Some techniques are based on link inspection which performs the operation on the basis of critical links surveillance [107], [2], [20]. The basic phenomena of these techniques are to inspect the links and if any flooding is identified, then perform the mitigation process. A common characteristic of the current LFA defense techniques is that they offer mitigation for the traditional networks [20-22], [23], [24]. Some authors used SDN testbed to perform the experiments [25], [26] or use SDN-based techniques to mitigate LFA. However, there is a lack of literature available that mitigate LFA on SDN, specifically control channel LFAs. Therefore, it is still a challenge to mitigate control channel LFA in SDN, to address this challenge, we propose CyberPulse an efficient LFA mitigation system to provide defense against control channel LFAs. This study comprised up of comprehensive background study of LFA in SDN and provides a framework named as CyberPulse to mitigate LFA. Finally, the implementation of the CyberPulse as a module in Floodlight controller is presented. CyberPulse offers three modules name as Link Listener, Flood Detection and Flood Mitigation Modules. Link Listener Module continuously monitors the link and inspects incoming traffic by utilizing a JSON-based pattern matching engine, which has the ability to perform the computation on real-time line-speed level.

Efficient pattern matching is a key enabling technology for the full realization of network traffic monitoring. The pattern matching engine represents the traffic signatures into JSON format and performs deep traffic inspection. It then compares the incoming traffic with the JSON signatures stored in the database and identifies the malicious traffic streams. The pattern matching is performed using Regular Expressions (REs). After carefully analyzing the network traffic using pattern matching techniques to identify malicious flows, we preset CyberPulse Flood Detection Module which employs comprehensive machine learning algorithms and deep learning-based Artificial Neural Network (ANN) technique to classify the traffic into benign and flooding traffic. We classify the network traffic using two experimental paradigms. In the first scenario, we use deep learning-based ANN backward propagation technique to accurately classify the incoming traffic and in the second experiment, we train the machine learning classifier using any given algorithms and the training dataset. The classified traffic is then transferred to the Flood Mitigation Module which mitigate the flood traffic by using NULL routing technique and enable the normal flow of operation of the network.

It is to be noted that CyberPulse is implemented as an extension module in the Floodlight Controller. Pattern matching technology is the first line of defense towards a network security solution implemented in CyberPulse. Therefore, we describe pattern matching technology in the following. Most of the IT infrastructure across the globe is virtualized and is backed by SDN. Therefore, SDN is the main building block for today’s computing powerhouses making it a prime target for different attacks.  SDN’s vulnerability to being attacked by LFA increases knowing the fact that there is a central controller that is capable of managing the whole network. This centrally controlled network management exposes SDN towards a variety of different attacks in general and LFA in particular. In this research, we mainly focus on LFA in relation to SDN on all layers (data, control, and application layer). Figure 1-3 demonstrate the effect of LFA on three layers of SDN. It has been illustrated that the adversary can manipulate bots which send low rate attacks flows in order to achieve LFA causing overwhelming damage to different layers/planes of SDN. The consequences of an attack are presented in the form of attack results in the figure. We explain the attack consequences on different layers of SDN in the following.

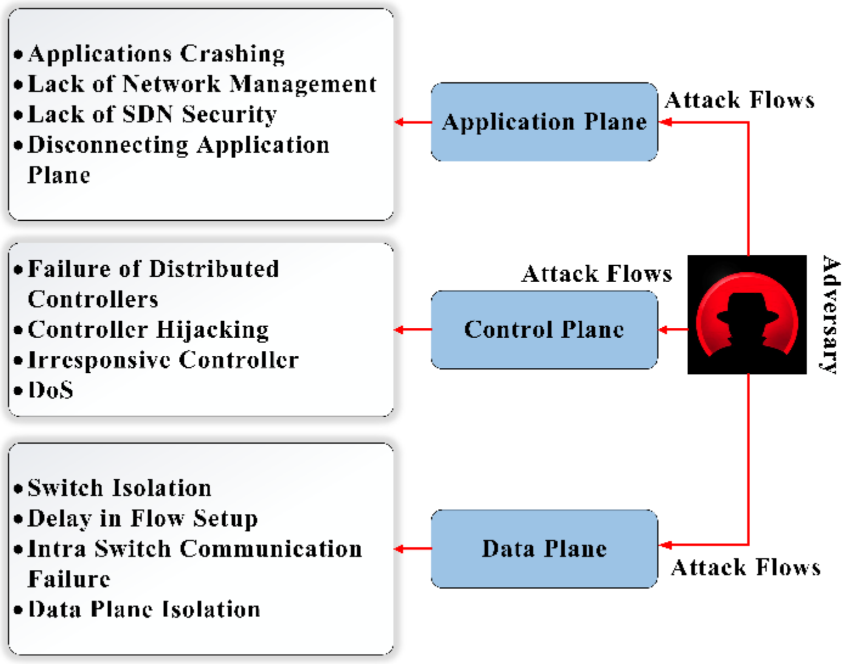


Figure 1-3 Effect of LFAs on SDN.

### *1.2.1 Application layer*

This layer contains the services and applications that make requests for the network functions from the data and control plane. LFA on application plane can cause applications running on this plane to crash which disrupts the normal flow of SDN. Network management applications are a critical component of the application plane. Network security is the prime concern of the application layer, LFA can bring security-related problems in the whole SDN. The crash of the application plane can destroy network operations and break all the services of SDN [27].

### 1.2.2 *Control Layer*

This layer works as a central control unit in SDN which is responsible for successful packets delivery from source to the destination [28]. SDN controller makes use of different interfaces to communicate with other layers and network elements like east, west, north and southbound APIs [28]. The most utilized APIs of the controller are northbound and southbound API. By utilizing the southbound API, the controller manages the communication with the infrastructure layer and other network devices. The northbound API is used to connect the controller with several network applications [29]. The northbound APIs don't use a standardized protocol as opposed to the southbound APIs that creates numerous security threats [30-32]. The east and westbound APIs manage the distributed controllers that are used to manage different portions of the network [33]. distributed controllers are installed in order to avoid a single point of failure or bottleneck [34].

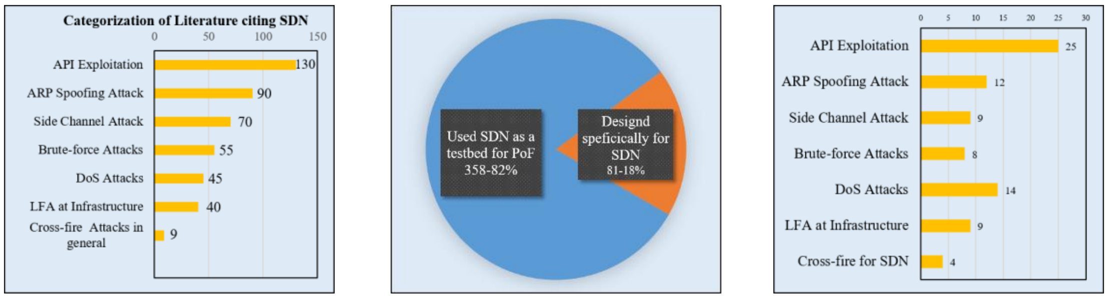


Figure 1-4 Survey of 439 research papers citing SDN and the attack types.

LFA on control plane can crash and disconnect distributed controllers from other planes provoking network failure. Additionally, flooding can cause DoS on the control plane, leaving controllers unable to fulfill legitimate requests. Previously, extensive research has been conducted in the field of DoS like [35-39] and exploiting fault-tolerant properties in controller [40-45]. Figure 1-4 provides a survey of 439 research papers citing SDN which demonstrates that only 18% of research papers have been conducted for specifically SDN other have used SDN as a testbed for the proof of concept.

### 1.2.3 *Data Layer*

Most of the earlier attacks on data plane were about overloading the flow tables of the SDN switches by inserting fake flows and depleting the memory [46, 47], [48]. In [49]

Sood et al. evaluate the performance of SDN switches by processing the incoming packets without the interaction of the controller. The attack on data plane [27] can severe consequences on the network because the data plane contains the core hardware to accomplish the communication process. Under LFA, the network devices on the data plane can get disconnected making the network services irresponsive. This attack involves flooding the infrastructure layer causing communication delays and foster performance degradation. Another severe result of such attacks on the data plane is disconnecting switches from each other which results in loss of flow rules and synchronization issues between coordinating switches [50]. LFA can also harm the flow rules installation process in SDN switches which causes delay or even disconnecting rule installation service and ultimately slowing the whole network. LFA can isolate the data plane [27] from SDN bringing the whole network down. LFA can have an overwhelming effect on SDN switches, as the switches cannot process the packets with their actual processing speed. The characteristics of LFA has been discussed in the following.

## 1.3 LFA Characteristics

Apart from the above discussion, we have performed an extensive literature survey on LFA and found the following characteristics.

- LFA can get selected links choked and hence disconnect a specific region from other regions [51].
- Detecting or mitigating the LFA is more difficult than defending the distributed DoS attacks [25].
- LFA is unique as it has two unique characteristics which make them drastically consistent in flooding specific links and interrupting the legitimate traffic.
- The first is the indirect strategy to attack a target server as the target of the attack in is different from traditional DDoS, therefore, it is complex to detect these attacks [3].
- The second is the that these attacks employ protocol conforming traffic which is indistinguishable from the legitimate traffic which lasts for more time and causes much damage to the underlying network.

With the above characteristics, it is harder to detect LFA and then propose any relevant mitigation strategies. Multiple mitigation strategies have been proposed previously in order to mitigate LFA [1, 20, 22, 52] including traffic engineering [1, 53, 54], link monitoring [2, 18, 23, 54] and SDN principle-based approaches [25, 26, 55]. In [50] Niyaz et al. have elaborated different attacks on SDN and their impact on the web services provisioning. Figure 1-4 attempts to categorize different attacks in the published research citing SDN. Despite its importance, LFA received the attention of about forty research papers. This figure highlights that only 18% of the works mentioned in figure were actually targeted towards SDN, while the rest used SDN for proof of concept. The bar chart in figure clearly describes that only nine research papers have so far considered LFA problem in SDN and that too mostly at the data plane. Different techniques have been employed to mitigate the LFA. The pattern matching technology is one of them, it provides the first line of defense against the LFA.

## 1.4 Pattern Matching Technology

The security of the SDN plays an important role in future generation computer systems. Numerous protection solutions have been proposed against the attack traffic. A lightweight pattern matching solution provides a first line of defense against the lethal security attacks. Pattern matching works in a way that it searches for a specific pattern of traffic in the network [56]. Modern network security solutions for examples NIDS, firewall, and IDS routinely deploy DCI techniques to ascertain malicious traffic. The contents of the packet are examined using patterns of matching malicious traffic to identify attack packets. It searches for the fixed sequence of bytes in a single packet. In most of the cases, pattern matching technology works by inspecting the packet whether it is associated with a particular service or destined/originated from a specific port.

Hence, this process helps in identifying the total number of packets to be matched and increase detection efficiency. The pattern matching has been used in the social security numbers, telephone numbers, zip codes, or any malicious activity detection in a sequence. The pattern matching has to fulfill two requirements, first the speed of pattern matching should be fast enough and secondly, they should be space-efficient to ensure the Service Level Agreements (SLA). Moreover, the security solutions are often provided as the middle-boxes at the Gbps links hence the pattern matching engines must be fast enough to meet the network requirements. The available pattern matching technologies suffer from space-time constraints. We implement a pattern matching engine for the security of the network traffic.

Table 1-1 Meta characters and their meanings.

| Symbol | Meaning |
|---|---|
| . | Any single character |
| * | Zero or more occurrence of the previous character |
| ? | Zero or single occurrence of the previous character |
| + | One or more occurrence of the previous character |
| ^ | Indicates negation or start of the line |
| [] | Defines character class e. g [A-Z] and [0-9] |
| $ | Matches the end of a line |

CyberPulse solution utilizes pattern matching in Link Listener Module to filter network traffic in SDN and to identify malicious traffic patterns. Network traffic can include wide-ranging of formats such as HTTP, Web, Synch traffic, data traffic and network packet captures. Pattern matching commonly occurs in the embedded device sensing the physical properties of the environment such as cameras, motion detectors, temperature, and light sensors or in the cloud where the data is aggregated.

DCI and intrusion detection and prevention (IDS/IPS) also rely on identifying malicious content against defined patterns. In all these cases, the language used needs to be expressive enough to parse a variety of patterns as well as efficient in terms of space and time requirements. Hence, there exist a number of ways to use patterns in different network traffic identification applications. In the past few decades, REs has emerged as the most appropriate way to define textual patterns as they are highly flexible as compared to fixed strings. At a first glance a RE seems to be an encoded piece of message e.g. *"^a {2, 4} b*{bc | ae} + [^d]"*. This lacks in human readability due to excessive use of meta-characters, which have special meaning when they are combined with the literal characters in the RE. At the same time, the flexibility present in REs is also due to the use of meta-characters and counting constraints. Table 1-1 gives a description of some of the well-known meta-characters that are mostly used in REs.

RE searching is a multistage process. First, a RE is generally parsed to obtain a tree representation. In the next stage, it is converted into an NFA (Non-deterministic Finite Automata) and then into a DFA (Deterministic Finite Automata) using the Thompson algorithm and DFA classical respectively. The DFA obtained through this method often comprises of redundant states and transitions that occupy large memory. This process of RE to NFA or DFA conversion is mostly known as the compilation phase. Moreover, a RE can either be compiled individually or a set of REs can be compiled together to generate a composite DFA. For *k* REs, a composite DFA has a processing cost of O (1) whereas it is O (*k*) for individual DFAs (as the processing cost for the DFA of each RE is O (1)). A composite DFA has the advantage of low processing cost over individual DFAs [57]. However, in certain cases, it becomes infeasible to generate a composite DFA for a given set of REs.

Most of the times this is due to the use of wildcards that result in DFA state explosion. Such complex DFAs not only consume time in their construction but also consume a large part of memory, which is evident from the experimental study given in this study. This problem is faced in both the cases of DFA creation either a group of REs is compiled or only a single RE is compiled [58]. Generally, the same pattern can have different RE representations but they may differ in their efficiencies. For example, we have a RE "*SEARCH\s+[^\n]{1024}$"*, which detects any input string with the occurrence of SEARCH at the start of the line, followed by one or more white spaces and then 1024 characters except for the return character.

Due to overlapping in this RE (overlapping will be discussed later in this chapter) caused by *"\s+" and "[^\n] {1024}"*, the resultant DFA will be of quadratic size. However, the same pattern can be represented efficiently by following the rule rewriting technique mentioned in [57]. After using the rewriting technique, the new RE "^SEARCH\s [^\n] {1024}$" results into a DFA, which is far more memory efficient as compared to the previous one. Thus, DFA implementation of REs become impractical in certain cases and one has to look for other options for the practical implementation. Many times, a new pattern could be written with a minor extension or modification to the existing one. Patterns, especially large ones required for big data parsing in IoT environments represented in RE format make this task difficult.

## 1.5 Pattern Matching Algorithms

There are many algorithms proposed in the past to minimize the size of DFAs that work by exploiting the redundancy in transition tables and states. Much of the current research work exists either on DFA minimization or in reducing the time of the pattern matching engine while matching the input stream against patterns [59], [60], [61], [62], [63], [64], [65], [66], [67]. In contrast, there is relatively less work available focusing on the compilation phase although it is the most critical phase of a pattern matching process. REs that have been expressed in a suboptimal way can lead to a large and overly complex compilation of patterns. It used to be the case that the potential problems of this phase were mostly overlooked because the compilation of patterns was performed offline very infrequently and subsequently stored for long-term use.

However, nowadays patterns using in-memory databases require frequent updates. In-memory databases have gained importance because of their high-performance with big data applications such as for Internet of Things (IoT). Consequently, it is no longer a good solution to store the pre-compiled DFAs in memory. Thus, the significance of the compilation phase needs to be highlighted so that this area of research can further be explored and the problems related to the online compilation of patterns can be addressed. This work considers RE as the root cause of many problems (discussed in next chapter in detail) and points out that there is no significant work performed in the literature for better representation of patterns or to improve the current RE notation. After discussing the pattern matching technology, we present the vulnerabilities in SDN towards LFAs.

## 1.6 SDN Vulnerabilities

In this research, we demonstrate with extensive experiments that how LFA pose threat to modern SDNs. We used the testbed comprised up of Mininet network emulator [68] and floodlight open-source controller version 1.2 [69]. We perform the simulation of the attack by employing low-rate traffic originating from multiple sources to create an effect of LFA on SDN. An application has been developed named as CyberPulse leveraging the application plane of the SDN. This application is based on three components including Link Listener Module, Flood Detection Module, and Flood Mitigation Module.

A JSON-based pattern matching engine is used to perform the pre-filtering of the traffic by comparing the traffic with the already stored patterns using a sophisticated representation of REs. The filtering of the traffic is necessary for Aml and humanized computing as the IoT generate a huge amount of traffic a pattern matching which can filter the traffic efficiently. Previous research in this paradigm stresses the use of hardware and specialized systems to provide defense against known attack patterns. However, there is a lack of research in the optimization of the pattern matching, RE as well as the compilation process of translating patterns into data structures. This study stresses the importance of developing efficient REs for efficient pattern representation. Therefore, we conduct an empirical comparison of multiple RE processing pattern matching engines and demonstrate with practical implementations that the compilation phase of the pattern matching consumes a lot of memory as well as take a lot of time in comparison with the pattern matching phase where optimization must be sought to optimize this phase.

A fundamental characteristic of SDN is that it provides separation between the control plane and data plane. This separation provides better management and flexible programmability of the network elements. OF is one of the widely used protocols for communication in SDN. Moreover, network services are developed as applications in the application plane which interact with the controller using northbound API. Along with certain advantages of centralized network management, there are many threats that leverage the centralized control capability of SDN. These threats include increased probability of DoS because of centralized control, flow table overflow, open programmability related issues, and DoS.

These threats have the ability to bring down the whole network and disrupt the service provision which increases the cost of service orchestration, in the same way, increase traffic delays. Network malfunctioning greatly exacerbates the infrastructure service cost, indirectly increasing the total network operating cost. Providing seamless network services have become a critical challenge nowadays [70], [71]. Hence, mitigating threats to the network performance and operation has become vital in order to meet network availability requirements

## 1.7 Motivation

The motivation for this study comes from the analysis of the LFAs which have emerged as one of the most devastating attacks during the past some years. In contrast to traditional DoS attacks targeting individual servers [72], LFA target critical links that connect to important servers [3]. LFAs are devastating in nature because they clog core links paralyzing and sometimes disconnecting the target network connected to the link. LFA is a growing threat to the network infrastructure and has attracted a lot of attention from academia and industry [1]. Mitigating LFA is a challenging task because of three important reasons: first arises due to the communication between bots and the public servers. This implies that the attack traffic is indistinguishable from the legitimate flows and sometimes unobservable when the target server does not belong to the victim's network. Hence, typical flow filtering techniques to screen-out malicious traffic become inoperative. Second, the attacked link and the target server's network are part of different Autonomous Systems (ASes) which are different from the source (attacker) AS. These AS have no visibility into each other's network activity and they never know that a link is under-attack. In this regard, inter-AS cooperation is necessary to mitigate LFA.

However, source AS lacks incentives to push them for cooperation because the collateral damage for them is negligible. Third, the attack persistence, as multiple bots send traffic to the target server which is not distinguishable hence traditional security measures becomes spurious and the attack becomes persistent which makes it difficult to mitigate. Traditional IDS/IPS countermeasures become invalid in this situation because the attack traffic may never reach the target server. The attackers send low-rate traffic to the carefully selected servers with a target of flooding the links connecting to the server. This action results in an increase in flood traffic on the links leading to the target server. Hence, legitimate traffic to the target server is interrupted.

Typical flow filtering techniques fail in this context because low-rate traffic cannot be detected by the rate limit threshold mechanisms. In Figure 1-5, the phenomena of LFA is depicted. An adversary is presented as sending traffic to the bots which further send traffic to the links towards the target server. Ultimately, the benign user will be unable to connect to the target server. In the start of LFA, the prober of the adversaries utilizes traceroute packets to create a link map of the network. Then the adversary calculates the attack-cost strategy and ascertains the number of bots that can occupy the links. Then the adversary sends TCP like traffic to the decoy servers to occupy the links. Due to the fact that bots deploy low-rate traffic, they perform LFA by remaining undetected. This sequence of operation results in the complete utilization of the link capacity and legitimate traffic will be unable to pass through the link.

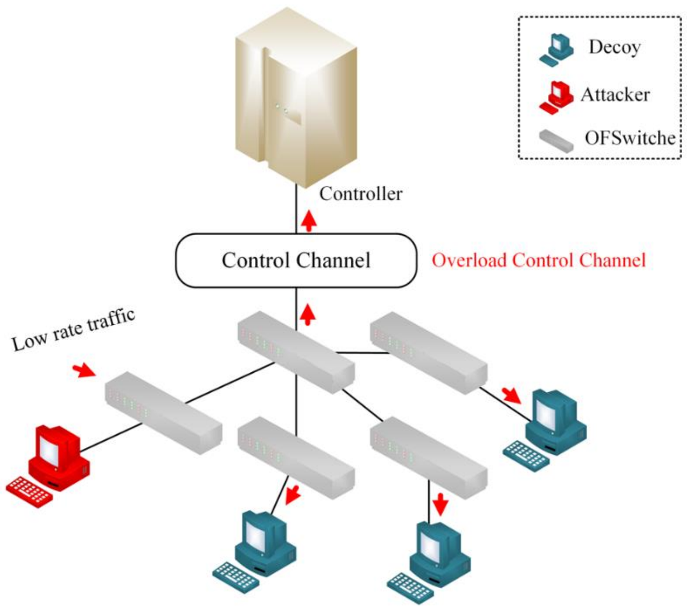


Figure 1-5 The setup of LFA on the control channel.

Due to these challenges, LFA has gained immense attention in the literature since they were first introduced by Kang et al. and Studer [4], [73], [4]. A typical category of techniques to detect LFA is to employ link inspection for network traffic measurement. The idea behind these techniques is to permanently inspect the link to detect any malicious activity [2], [20]. LinkScope [16] utilizes both hop-by-hop and end-to-end network measurements to identify the occurrence of LFA. Such techniques help in diagnosing and localizing the attack links and facilitating the countermeasures against LFA.

Some available techniques employ link obfuscation to provide fake links to the adversaries hence making it hard for the adversaries to locate the target link. [2], [74], [19]. Most of these techniques use graph metric-based measures to identify the vulnerable links and subsequently, present fake links to the attacker to cater to the severity of the attack on the weak links. Even though LFA has gained huge attention from researchers, most of the existing literature is unable to distinguish the attack taxonomy on SDN versus traditional networks.

Categories of defense against LFA can be divided into three broad types including, link inspection, traffic engineering, and SDN-based approaches.  Link inspection techniques rely vastly on the network statistics measurement and then utilizing these measures to analyze the occurrence of the attack. Some of these techniques utilize the traceroute packets to increase measurement methodology to detect and mitigate the sources of LFA. Traffic engineering-based categories rely on multiple attackers and defender interactions in the network to expose the LFA sources. Techniques like host dropping or NULL routing can be further utilized to mitigate sources of LFA. The third type of categories relies on the centralized control mechanisms proposed by SDN to identify and mitigate the malicious sources. These techniques stress the controller-based measurements to inspect the flows in the network, and then utilize SDN principles to increase the network connectivity and alleviate the intensity of LFA.

SDN has become one of the most vastly adopted network standards and has attracted a lot of attention from academia and industry. It is a building block for most of the current networks, hence, securing SDN will ensure the security of huge network infrastructure all over the world. Control plane is the most vital component of SDN which handles all the network operating decisions. Data plane is regarded as the forwarding plane which forwards the traffic according to the rules installed by the controller on data plane devices. Most of the existing literature have been unable to distinguish LFA on SDN and traditional networks [20], [22], [21], [23], [1]. Some authors used SDN testbed for proof of their research model, however, there is a lack of research available which specifically address LFA in SDN paradigm. So, providing defense against control channel LFA is one of the prime concerns to secure current internet.

A recent solution providing defense against flooding attacks in SDN called FloodDefender [75], leverages controller capability to perform network measurements and flow rule management characteristics to defend against flooding attacks. It also employs table miss-oriented techniques and packet filtering to mitigate DoS attacks. Similarly, FloodDefender distributes the load to the neighboring switches to protect it from clogging. However, flow filtering and table miss-based techniques become invalid in case of LFA which uses low-rate legitimate traffic to flood the network. FloodShield [76] is also one of the current technique to cater DoS, however, they are unable to provide mechanisms against low-rate traffic. Moreover, the currently available LFA defense techniques lack in providing solutions that work on real-time speed. As the current IT infrastructure has become much critical due to the heavy reliance on online resources, providing defense in real-time has become a vital challenge. In the same way, as network surveillance techniques are becoming critical, there is also a need to provide a solution that provides real-time network information to the network administrators. Similarly, the available solutions focus only on detection, mitigation or have narrow applicability, despite we stress the need for a holistic approach to this problem and offer a comprehensive framework which encompasses several top-notch components including, real-time network surveillance, LFA defense at line-speed, high accuracy of defense, and less overhead in terms of implementation. We propose CyberPulse a security framework to comprehensively secure SDN against LFA. We make the first effort to deal with LFA in SDN by proposing a robust solution. CyberPulse uses an algorithm-agnostic approach which leverages the facility of training ten ML algorithms. We propose two novel mechanisms to stimulate the cooperation between ML and SDN environment.  We have built a prototype solution after a thorough analysis of LFA taxonomy with reference to SDN. CyberPulse provides a secure SDN environment which inspects the network traffic on a real-time basis and mitigates LFA.

The first mechanism is to separately train the ML classifier on various classification algorithms. The basic idea is to utilize ML algorithms to perform the training on any provided dataset for flood attacks classification. CyberPulse has the capability to provide defense against LFA online speed and provides real-time network status information to the intended users. The second is a prototype solution, CyberPulse which performs the surveillance operation by collecting real-time network statistics. It continuously performs ML classification against the trained ML algorithms. CyberPulse employs algorithms for testing from the provided ten algorithms with multiple confidence intervals. If CyberPulse detects an LFA, it mitigates it by dropping the malicious flows. This framework is capable of providing multiple real-time network status graphs which makes it possible to perform the surveillance in a very interactive manner.

### *1.7.1 Research Questions*

This study aims at mitigating link flooding attacks against modern networks. The critical assertion in doing this is that the current network services are increasingly being provided using software. This softwarization brings new challenges of security, reliability, fault-tolerance, and seamless end-to-end delivery. In this regard it is very crucial to secure current networks against different attacks to ensure efficient service provisioning in modern networks. We systematically evaluated the concern of different attack possibilities and came up with the following research questions in this research.

**RQ1.** What are the current vulnerabilities in the current networks due to the softwarization of the network services?

**RQ2.** How these vulnerabilities can be exploited by the adversaries to attack the current networks?

**RQ3.** How different are SDNs as compared to the conventional networks, and can the established mitigation methodologies still be used effectively in SDNs as well?

**RQ4.** What are the major attack types with potential to disrupt SDNs, which are still unaddressed in the literature?

**RQ5.** What are LFAs what are its characteristics, how they evolved from different attacks.

**RQ6.** What is the current research progress against LFAs in SDN and what are the critical approaches that can be adopted to mitigate LFAs.

**RQ7.** What are the current LFA mitigation approaches, and how the available ML techniques can be effective in mitigating LFAs at real-time?

**RQ8.** What are the future research paradigms that must be considered while addressing LFAs in the future networks?

## 1.8 Research Contributions

Highlights of some of the important contributions of this research are given below:

1. Extensive experimentation has been performed to demonstrate the challenge and impact of LFA on SDN. We illustrate the LFA related vulnerabilities in SDN and demonstrate how the performance of SDN is degraded in the absence of precautionary measures and indicate how ultimately LFA can bring down the whole network.
2. A JSON-based pattern matching engine has been developed which is more expressive in network traffic pattern representation and efficient in LFA traffic filtering. We conduct detailed experimentation to demonstrate the effectiveness of the JSON-based pattern matching engine. Moreover, we illustrate the need for a real-time RE compilation phase and demonstrate how our approach is efficient in real-time traffic filtering.

3. A detailed implementation has been performed where the design of an active traffic inspection and defense against LFAs has been demonstrated. It is an interactive solution which can provide real-time defense against large-scale LFAs.
4. We also provide a novel algorithm-agnostic approach of CyberPulse which offers extensive control over the defense system. CyberPulse introduces a newer trend in the ML-based network security which liberates the system from using only a specific ML algorithm or training dataset.
5. The performance evaluation of CyberPulse has been illustrated using ten state-of-the-art ML algorithms. We employ a multitude of network traffic settings and demonstrate the efficiency by practical experiments that it is efficient, induces low overhead, and interactively provide defense against large-scale LFAs.

## 1.9 Organization of the Study

We initially present a comprehensive literature review of LFA and its mitigation techniques in relation to all layers of wired and wireless SDNs. Subsequently, a comparative analysis of mitigation techniques is presented with pointers to suitability for each of a given SDN type. We categorize the LFA and classify existing countermeasures based on where and when they prevent, detect, and respond to the LFA. Similarly, the need for robust pattern-matching technology to provide a first line of defense against LFA has been provided. Finally, we present a security framework that comprehensively mitigates LFA on the control channel of SDN. This study has been organized as follows.

Chapter 1 provides an introduction of the study where the problem of LFA on all SDN planes has been extensively discussed. Chapter 2 encompasses the literature on SDN, LFA, pattern matching technology, and its impact on LFA defense. Chapter 3 formalizes the problem and discusses the components of the defense system against LFA. Chapter 4 provides a detailed analysis of the DCI technology to defend against any abnormal network traffic. Chapter 5 discusses the feasibility to use ML as a defense against LFA, we provide ML tools used in the classification of LFA traffic and compare different algorithms. Chapter 6 illustrates the experimental evaluation of CyberPulse and discusses detailed evaluation results. Similarly, Chapter 7 provides the conclusion and presents current issues and future research directions.

## 1.10 Chapter Summary

In this chapter, we provided the preliminary knowledge on LFA and SDN. Moreover, we discuss the architecture of SDN and its components. The additional benefits of SDN due to its abstract layer representation has been elaborated. It has been illustrated that how SDN can benefit efficient traffic forwarding and control due to its centralized management. Usually, the centrally singular controller can have multiple other distributed controllers which provide an abstraction of a centralized controller. We discussed different attack types on SDN planes, specifically how they can be attacked by LFA.

We illustrate how the control channel can be attacked by a variety of ways. The motivation for this study comes by the analysis of the LFAs where there are a few mitigation schemes available that too are developed for the traditional networks. Moreover, the centralized control also poses vital security bottleneck where the controller can be disconnected from the rest of the network. LFA can congest the link between the control plane and the data plane. As the communication in SDN is based on the availability of this link, any attack can prove to be a degradation of the performance of the network. In ultimate conditions, it can disconnect the control plane from the data plane and bring the whole network down.

We elaborated the DCI principles and how it can provide a first line of defense against LFA. Moreover, we provide different algorithms of the pattern matching technology. In the next chapter, we will present the literature review on SDN, LFA, and its variants. We will especially focus the discussion on how the control channel can become a bottleneck in the presence of LFA. We will explain the vulnerabilities of LFA in SDN planes where LFA can pose damaging effects. We will explain the vulnerabilities of SDN in all its variants including, Software Defined Mobile Networks (SDMN), Software Defined Wireless Networks (SDWN), Software Defined Wireless Mesh Networks (SDWMN), and Software Defined Wireless Local Area Networks (SDLAN). Moreover, we explain the weaknesses of SDN where LFA can be critical.

Network flooding is essentially a DoS which is devised to bring a network or a service down by using a large amount of traffic. These attacks occur when the service is brought down in such a way that the legitimate connection requests cannot be completed by the network/server. DoS and DDoS are one of the most intimidating threats which the modern networks face causing an hourly loss to be around $40K an hour [77]. LFA is as a new type of DDoS attack which can degrade or even cut off network connectivity of a target area.

This attack employs low density legitimate flows to flood the certain links, making the malicious flows extremely hard to be distinguished by traditional defense mechanisms [78]. In order to devise any mitigation technique, it is important to understand LFA's detailed background, types and variants, and what makes it so important in the context of SDN. And for this very reason, this chapter summarizes dispersed background studies and puts in a perspective of LFAs, and then discusses the evolution of LFAs and the variability of these attacks which give rise to different LFA types. It also presents a discussion around what makes LFAs so lethal, especially in connection to unique nature of SDNs. With the intention to understand the effectiveness of LFA threats against various SDN deployments, we categorize prevailing LFA research [79] on the basis of the mechanism used for the detection and mitigation in the context of SDN. Additionally, we present a taxonomy of current research to defend against LFA using performance metrics.

In order to help answer the presented research questions, after covering SDN preliminaries we carry out a detailed discussion of LFA and feasibility of mitigation techniques, and its implementation on all the SDN variants. The aim is to help reader understand the problem along with its intensity, and then gradually steer to a logical solution. The intent is getting research questions 1 to 6 answered, hence the details and multiple comparative analysis and the associated insights.

## 2.1 SDN Preliminaries in Context to LFA

In SDN, the network behavior is controlled centrally by using Application Programming Interfaces (API). SDN adds intelligence to the network management by decoupling the packet forwarding control structure from the switching hardware [9, 12, 33, 80]. Network traffic is forwarded without changing any setting of individual switches or routers by using this approach. This novel network architecture consists of three planes namely, data, control, and the application plane as depicted in Figure 2-1.

The data plane also commonly known as the infrastructure layer contains forwarding devices, responsible for packet forwarding, the control plane is responsible for the configuration of the data plane and the application plane receives customized network requests from that are implemented by network programmers in the form of applications [81, 82]. LFA can severely affect the SDN architecture by attacking the links between infrastructure layer and the control layer and disconnecting it from the rest of the network. This attack disconnects the entire infrastructure layer from the SDN. SDN controller's intelligent logic resides in the control layer that makes the network infrastructure programmable. This is an area where we see a large level of commercialization whereby network vendors are offering customized solutions for the SDN controllers and framework. The business logic primarily resides here and is responsible for controlling network infrastructure and fetching and maintaining different types of network information like state, topology, and statistics. SDN controllers contain all the necessary logic for network management that includes routing, security rule set for firewall, DHCP, clustering, DNS, switching and routing. In parallel to three layers, there are two interfaces for communication between the different layers, i.e. northbound and southbound interfaces.

The northbound interface is used for communication with the upper layer and in general, it is accomplished through the REST API of SDN controllers. The southbound interface is used for communication with the lower layer, known as an infrastructure layer, which includes network elements and implemented using OF [12], Netconf [83], and Ovsdb [84], etc.

The application layer is a relatively unexplored domain to develop as many innovative applications can still be developed that fetch important network information such as traffic statistics, state, and topology. There are several possibilities of applications which can be developed including those related to network automation, configuration, and management, monitoring, troubleshooting of network policies and security. SDN has revolutionized the way traditional networks were managed since it dramatically reduces network operating cost by using inexpensive switches that can perform automated network functions. Different network configurations can be tested without disrupting the actual network.

Due to the centralization of the Forwarding Information Base (FIB), optimal routes can be identified for seamless traffic flow in the network. SDN can filter the packets as they enter the network. Data plane switches can act as firewalls and are able to redirect traffic flows to security controls at higher-level layers. In contrast to these advantages, centralized control of SDN makes it vulnerable to multiple attack types intended on all layers of SDN including flow table overflow attacks [13, 46, 85-93], control channel flooding attacks, link fabrication attacks [15], and controller failover attacks. SDN implementation is also a very challenging task, as it requires a completely dynamic network infrastructure.

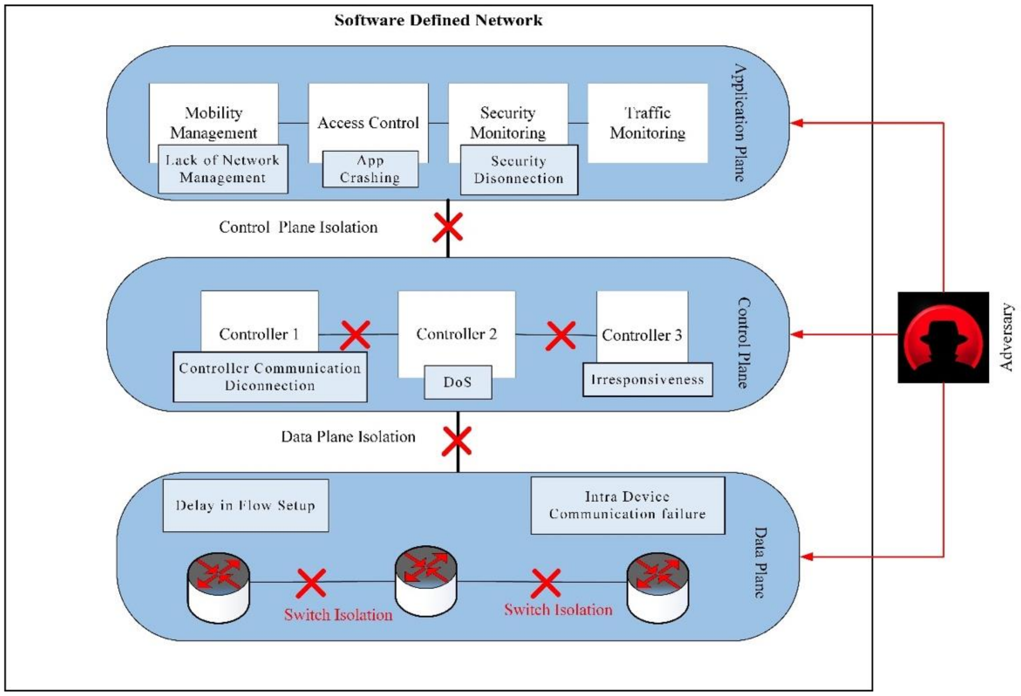


Figure 2-1 Effect of LFA on all SDN planes.

The consequences of LFA on all the three planes of SDN is presented in Figure 2-1 where an adversary is sending LFA traffic on all SDN planes. SDN application plane enables configuring the switches to forward customized traffic, integration of networked-applications and administering controllers. The application plane contains the applications which manage the whole SDN, like mobility management, access control, security monitoring, and traffic monitoring. Although the complexity of launching LFA on application plane is immensely high as it requires authentication, however, it would have a disastrous impact on the network if it gains the privileges and passes the authentication. The impact of LFA on SDN application plane is severe because it enables the applications to run smoothly, the whole network suffers if it happens to be under attack. LFA on application plane invokes a complete or partial disruption of these management modules from the whole network. Every module in the application plane performs some specific and specialized operation that is critical for the seamless operation of SDN. Flood traffic in application plane results in network security applications to crash and hence it is a security vulnerability in SDN. LFA can also disrupt the traffic running from application plane to the control plane in severe conditions the link can be disconnected. The application plane holds crucial network management policies like requesting network functions from other planes and building an abstract view of the network by requesting information from the controller. LFA has the potential to create severe network management issues by attacking this plane.

In the control plane, there are distributed controllers which coordinate each other to accomplish the network control functionality. The purpose of deploying distributed controllers is to set up a backup mechanism, as the controller is the backbone of SDN, the secondary controller can take charge in case of failure of the primary controller. LFA can disrupt communication between the controllers that work synchronously to provide network control functionality. As the controller is the critical component of SDN, it can be attacked by DoS which creates hindrance to successful network operation and may cause complete network failure in extreme LFA conditions. It is imperative from the SDN design that if necessary preventive action is not taken against LFA, the entire network can be brought down by the adversaries. Therefore, it is important to bake-in a higher level of resiliency.

The data plane is the core working element in SDN as it forwards the packets towards the destinations. The data plane switches, routers, and other hardware components forward packets according to the rules provided by the control plane. When a new packet arrives at the switch, it inspects the rule to forward the packet, if a switch is unable to find a flow rule for a packet it asks the coordinating switches about the packet and forwards it in case a rule is available in the flow table. LFA can cut-off the intra switch communication by flooding the links between the switches which results in rules updating problem and disruption of the communication between coordinating switches. Data plane contains the hardware to forward the packets to the desired destinations. It can be attacked in a variety of ways using LFA, which incurs devastating effects on SDN. A flow rule is added according to the packet destination, LFA can isolate the forwarding devices, which affects the rule setup in data plane switches.

## 2.2 DoS Attacks and Relevance

One of the most lethal attacks in modern networks is DoS. In this attack, an attacker employs TCP/UDP packets to flood the target server and intends to drop the services or a target server. These attacks shutdowns individuals services and machines which render vital communication and revenue loss.  The types of DoS attacks are as follows.

- Buffer overflow attacks
- Ping of death (POD) attacks
- SYN flood attacks
- Teardrop attacks

In the DDoS attacks, multiple attackers from different locations send attack packets to a target server. Using distributed taxonomy to attacks a target server brings the target down easily as the victim will not be able to service such a huge number of simultaneous service requests. The attacker creates an army of compromised botnets to attack a target server. Following are the main categories of the DDoS attacks.

- Volumetric Attacks
- Fragmentation Attacks
- TCP-State Exhaustion Attacks
- Application Layer Attacks
- Link Flooding Attacks

LFA as a main point of discussion here, has a prominent characteristic being able to use legitimate traffic to flood specific links –making it difficult to be detected and mitigated. Therefore, it is important to provide a taxonomy of the LFAs and their effects on different variations of SDN.

## 2.3 Link Flooding Attacks

LFAs were first proposed in 2013 as crossfire attacks. They are different from the traditional DDoS attacks in a variety of ways. In a DDoS attack, multiple compromised hosts send a huge amount of flows comprising of higher traffic rate to choke a target server. Hence, they can be detected by rate limit detection techniques. However, LFA uses low-rate traffic in a stealthy way which is difficult to be detected by rate limit detection mechanisms. On the other hand, the attacker uses legitimate hosts to send these flows hence, the detection of LFA portrays a greater challenge. The crossfire attack evolved further into different forms. We discuss some of its variations as follows.

### 2.3.1 *The Crossfire Attack*

The crossfire attack is more sophisticated and stealthier as compared to its variants such as coremelt attack [79] and is adaptable to route shifts and avoids triggering alarms for potential attacks by changing the target links after a specified time interval set by the attacker. ISP collaboration is an essential requirement for any effective defense using modern security tools. Therefore, all these factors need to be considered when developing a mitigation solution.

Furthermore, crossfire attack is the most critical LFA due to its low detection rate and indirect nature. This attack is carried out using bots as demonstrated in Figure 2-2, which send low-rate legitimate traffic to the selected decoy servers that are not the target servers but are situated on the path of the target. Sending low-rate traffic to decoy servers chokes the links connecting the target server with the rest of the network. The sources of the crossfire attack are undetectable by any targeted servers since they no longer receive any messages or flood traffic [3].

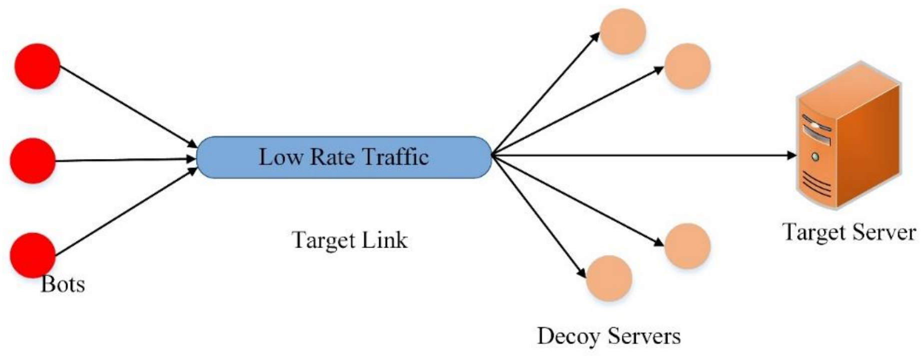


Figure 2-2 The crossfire attacks (the figure at [3] is expanded and redrawn).

It is very difficult to differentiate between legitimate users and malicious bots as both use valid IP addresses [3, 94]. The adversary starts by constructing the network profile by sending traceroute packets to identify the target server and critical links to attack. It then selects the decoy servers and computes the bot-decoy pairs that are sufficient to perform the flooding. Finally, it sends low-rate legitimate flows from bots to decoys, hence, all paths to the target experience flood traffic, as a result, the legitimate traffic is blocked. Crossfire has the potential to attack modern SDN in a variety of ways using the characteristics of being indirect and employing low rate traffic. One such scenario can be control channel LFA.

Table 2-1 illustrates the severity of crossfire based on its low detection rate as compared to other variants. The crossfire attack can disable network links by flooding them with attack traffic.

Table 2-1 Comparison of different LFA techniques.

| **Design Goal** | **Coremelt Attack** | **Spamhaus Attack** | **Crossfire Attack** |
|---|---|---|---|
| Independent bot distribution | X | Not a goal | √ |
| Indistinguishable from legitimate flows | √ | X | √ |
| Resilience on needed flows only | √ | X | X |
| Attack persistence | Low | Low | High |
| Use of specialized bots | X | √ | √ |
| Scalable for N Servers | X | X | √ |

Crossfire attack uses legitimate IP traffic instead of using spoofed IP addresses which can easily be distinguished. It then sends legitimate packets to publicly accessible servers (decoy servers). So, these packets keep on flowing without any interruption, flooding the target link which is situated among these servers. Lastly, it transmits low bandwidth flows from each bot individually, these legitimate flows then cumulatively flood certain links in the network without being detected.

The details of Spamhaus and Coremelt attacks have intentionally not given here in the interest of space and to keep the focus. Interested readers can refer to [3, 4, 5, 95, 101].

## 2.4 SDN in the Context of LFA

As established in the sections above, an emerging threat targeting internet infrastructure – LFA has gained a lot of attention in the literature during past several years [51], [96], [97, 98], [99], [88], [100]; but mostly in the context of traditional networks. LFA was initially proposed by Kang et al. called with the name of crossfire attack which has the ability to disconnect target links without being detected [3, 73]. We broadly categorize LFA techniques into three groups: 1) traffic engineering-based, 2) links inspection-based, and 3) SDN-based approaches.

Traffic engineering have broadly been adopted by researchers to mitigate LFA. In such a technique, Takayuki et al. [22] employ increase in traceroute packets volume-based measurement to detect LFA. Christos et al. [24] present a relational algebra-based technique which employs the attacker-defender interaction to attack and balance the network traffic. This continuous interaction exposes the attacker's identity and can be easily eliminated. Dimitrios et al. state that the defender reroutes the traffic in case of an attack, the defender recalculates the critical paths and send attack traffic again. During the multiple interactions between attacker and defender, the malicious sources are identified and can be mitigated. Virtual networks-based dynamic resource allocation strategy has been proposed by [37], however, the limitation with this technique is that it relies on the security of the virtual layer, which becomes vulnerable in case if the layer is exposed.

Link inspection techniques employ target link surveillance to inspect any malicious activity on the target link [102]. The link obfuscation technique proposed by Wang et al. [103] presents fake links to the adversaries which makes it harder for them to create a true link map of the network. LinkScope [16] inspects network links on hop-by-hop and end-to-end basis and reports any malicious activity observed on the link. Min et al. [2] propose SPIFFY technique which employs an increase in network bandwidth on a temporary basis and measures flow statistics before and after the bandwidth expansion. The legitimate flows tend to adapt to the bandwidth expansion, alternatively, malicious flows will result in ending up by consuming all their allocated bandwidth and hence will be easily detected. SDN HoneyNet proposed in [19] performs the graph-based statistics measurement on the network and identify the links with the lowest betweenness centrality that are prone to become bottlenecks during LFA. SDN HoneyNet [19] then provides fake link maps to the adversaries that will be unable to ascertain the critical links on the network. Collaborative Defense (CoDef) [20] portrays the concept of AS collaboration where the links that are not under attack can collaborate and the legitimate traffic can be rerouted towards those links for successful network operation.

Some researchers utilized SDN-based techniques to observe all the switches and their corresponding flows to avoid LFA. Peng et al. [55] propose a flow table inspection technique which inspects the flow tables when the resource utilization ratio of flow is not normal and employs bloom filtering technique to detect the adversaries. Incremental SDN deployment technique upgrades the routers to SDN switches to increase the network connectivity and employs traffic engineering principles to balance the network traffic [25]. SDN-based traffic maneuvering technique obfuscates the links making it difficult for the adversary to create true link map, hence avoiding the attack [26]. Xiaobao et al. [17] propose a cost incentive approach for the source and destination AS domain in order to cooperate to alleviate LFA. The source and destination of LFA remain in different AS domains, hence, the initiating AS domain have no idea of how its traffic is affecting the destination AS. So, a cooperation between the two AS is required to effectively detect and mitigate LFA, however for the source AS to cooperate with destination AS, an incentivized mechanism can be provided to encourage cooperation.

The above-mentioned techniques suffer from low detection accuracy, adaptability problems, leakage report, and pose more overhead to the network. ML-based solutions have the capability to learn from the historical flooding behaviors and detect the characteristics of flooding flows. As ML-based techniques have strong learning ability and defense efficiency, these techniques can be applied for LFA detection and defense. ML techniques have been vastly used by researchers for network security [104], [105], [106], [107]. Aapo et al. [108] propose an ML-based technique which "combines normal traffic learning, blacklists integration, and elastic capacity invocation to provide effective load control, filtering, and service elasticity during an attack". The external blacklists are obtained from existing IDS. Fairuz et al. [109] propose an anomaly detection-based approach to evaluating malware detection with ML classifiers.

They used two publicly available datasets which include, Malgenome and a self-generated dataset labeled by the experts. They evaluated their classifier using six ML algorithms including Bayes, Random Forest (RF), Multi-Layer Perception (MLP), J48, and K-nearest Neighbor (KNN). Their results illustrate that Bayes and RF performed well as compared to MLP. In [110] authors "propose a DoS attack detection system on the source side in Cloud, based on ML techniques. This system leverages statistical information from both the cloud server's hypervisor and the Virtual Machines (VM) to prevent network packages from being sent out to the network".

Authors considered multiple attack scenarios for evaluation including SSH [111] brute-force attacks [112], ICMP flooding attacks [113], DNS reflection attacks [114], and TCP

SYN attacks [99]. A model is trained using the training dataset and ML module is utilized to update the pre-trained module [115]. They used multiple classifiers for evaluation including Logistic Regression (LR), SVM linear kernel, SVM, Radial Basis Function (RBF) Kernel, SVM Poly Kernel, Decision Tree (DT), Naïve Bayes (NB), RF, K-means and Gaussian Expectation Maximization. Flow-based IDS for SDN was proposed in [116]. In this technique, an IDS "periodically gathers statistical information about flows from SDN" OF-switches [12] and perform the analysis and detects the intruders by utilizing the collected traffic features. They perform the classification using SVM, DT, Bagged Classifier, RF and KNN. In [76], authors propose a deep learning-based approach for DoS defense.

LFA if successful could potentially take entire countries off the Internet. Particularly, LFA: i) uses legitimate rather than spoofed source IP addresses to send traffic; these addresses are much harder to filter, ii) sends legitimate packets to publicly accessible servers, and iii) transmits low-bandwidth flows from each bot individually; these flows together then bring down certain links. In the next section we discuss the impact of LFA on different SDN variants.

The effect of LFA and the necessary mitigation strategies in traditional networks are different from SDNs. Therefore, it is important to study different variants of SDN and cross compare the features that make SDN unique and vulnerable for attacks. However, in the interest of the allowed space, these details are not presented here, and the readers are advised to refers to [117-128, 130,311] for in-depth understanding of software defined wireless networks, and [132-143] for software defined mobile networks. Research at [144-152] can be consulted for software-defined Wireless Mesh Networks and [129, 153-156] can be referred to for software defined wireless local area networks study.

## 2.5 Link Vulnerabilities in SDN

Majority of the earlier work [50] on SDN defense mitigates attacks that in directly cause a DoS [157, 161, 162], it includes threats like flow table overflow [157-159] and bandwidth consumption attacks [92, 157, 160]. For example, in [163] authors propose a framework for mitigation of control plane saturation attacks. [157] devises a polling strategy from the data plane to control plane for analyzing the control flows that come across for new communications to defend against control plane saturation attacks. A proactive flow rule analyzer and packet migration mechanism has been proposed to defend against control plane saturation attacks in [164]. In [36] authors present a list of comprehensive vulnerabilities in SDN where an attacker can potentially gain access to SDN and attack the network infrastructure.

From all the above discussion it is pertinent to say that previous work has been performed to secure, specifically data plane or the control plane leaving behind the most important link capable of driving the communication of the whole network. In case of LFA, the link between controllers to data plane can become a bottleneck for the successful operation of the network.  Any disconnection in this link can block the information from passing between the two planes, resulting in a complete failure of the network. In this context, our study has great importance in mitigating attacks on the control channel to secure modern network infrastructure. This work directly contributes to overcoming the weakness of SDN towards vulnerablility to LFAs.

Most of the global companies are transferring their infrastructure to the cloud. According to the Gartner survey [165] Amazon, Microsoft, and Google have virtualized 100% of their global computing facilities. This demonstrates that more and more companies are transferring their infrastructure from hardware systems to software [166]. SDN is becoming more and more prevalent and replacing traditional networking schemes. However, many threat vectors have been identified that can impact SDN architecture [118, 167-171]. SDN-based networks provide the ease to adapt to changing business requirements. Since SDN is a relatively recent concept, new threat vulnerabilities are being identified with the time.

We identify a new threat that can flood the link on the control channel of SDN. If this happens in a network, the control channel link can get chocked, making the controller irresponsive and not able to manage the traffic at all. The controller is the brain in SDN, it installs forwarding rules in the switches to control the traffic forwarding. In an instance of a successful attack on control channel link, the rule installation procedure cannot be performed and the entire network would go in an undesired state. Meaning, the services provided over the network would come to a stop, affecting the network services on the cloud and causing the cloud infrastructure to become totally unavailable.

## 2.6 Control Plane Limitations

After an elaborated discussion on LFA and its impact on SDN, we summarize the following weaknesses in SDN.

- The control traffic contains the most vital information for SDN because it performs all the critical tasks of flow rules installation, network configuration, and optimum path selection. The prime concern of an attacker would be to disconnect the controller path links to make it ineffective.
- Data plane is the main traffic maneuvering unit which is dependent on the controller communication where it derives all the information through the hardware devices, attacking data plane devices result in disruption of the services of the whole network.
- The control channel is a critical resource because all the control information passes through this link, and this is only link where any congestion invokes traffic delays.
- The adversaries can take control of this link by implying hard to detect LFA and congest the link in a way that it can become irresponsive resulting in disconnection of the whole network.
- The control channel has the tendency to be chocked by flood traffic hence, there must be a link-scanning mechanism that can check whether the congestion is caused by the legitimate traffic or due to the adversarial flows.
- Since the control channel doesn't have any flood detection and mitigation strategy, there is a strong need to imply link congestion detection and mitigation mechanism as this link is crucial for the whole network as well as a favorite target for adversaries.

The control channel in all the network types including software defined wireless networks can have serious challenges if attacked by LFA. The control channel communication can stop the interaction between the network devices and the control plane which will ultimately bring the whole network down. So, it can be derived that all the SDN variants are equally able to be attacked by LFA. In Figure 2-3 the control channel is presented that is being attacked by LFA, this can be disconnected resulting in communication failure of control plane with network devices. We discuss the prevailing LFA mitigation techniques.

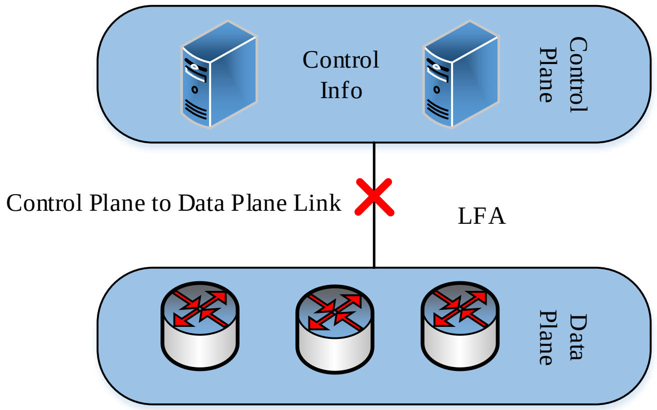


Figure 2-3 Controller communicating data plane devices.

## 2.7 Review and Analysis of Generic LFA Mitigation Techniques

In this section, we perform an in-depth analysis of the available LFA mitigation techniques. Existing work in this domain is more towards mitigating LFAs in traditional networks, however, a summary of the techniques may be useful to devise a solution for SDN. And some works which do site SDN in LFA mitigation work, in fact use SDN as a testbed to demonstrate the effectiveness of their techniques formulated for traditional networks. We compare existing techniques and categorize LFA mitigation according to the type of mitigation method used. Literature review presents three broad categories of LFA mitigation techniques are listed below.

- Traffic engineering principles-based mitigation techniques
- SDN principles-based approaches
- Link obfuscation-based techniques

### *2.7.1 Traffic Engineering Principles-based Approaches*

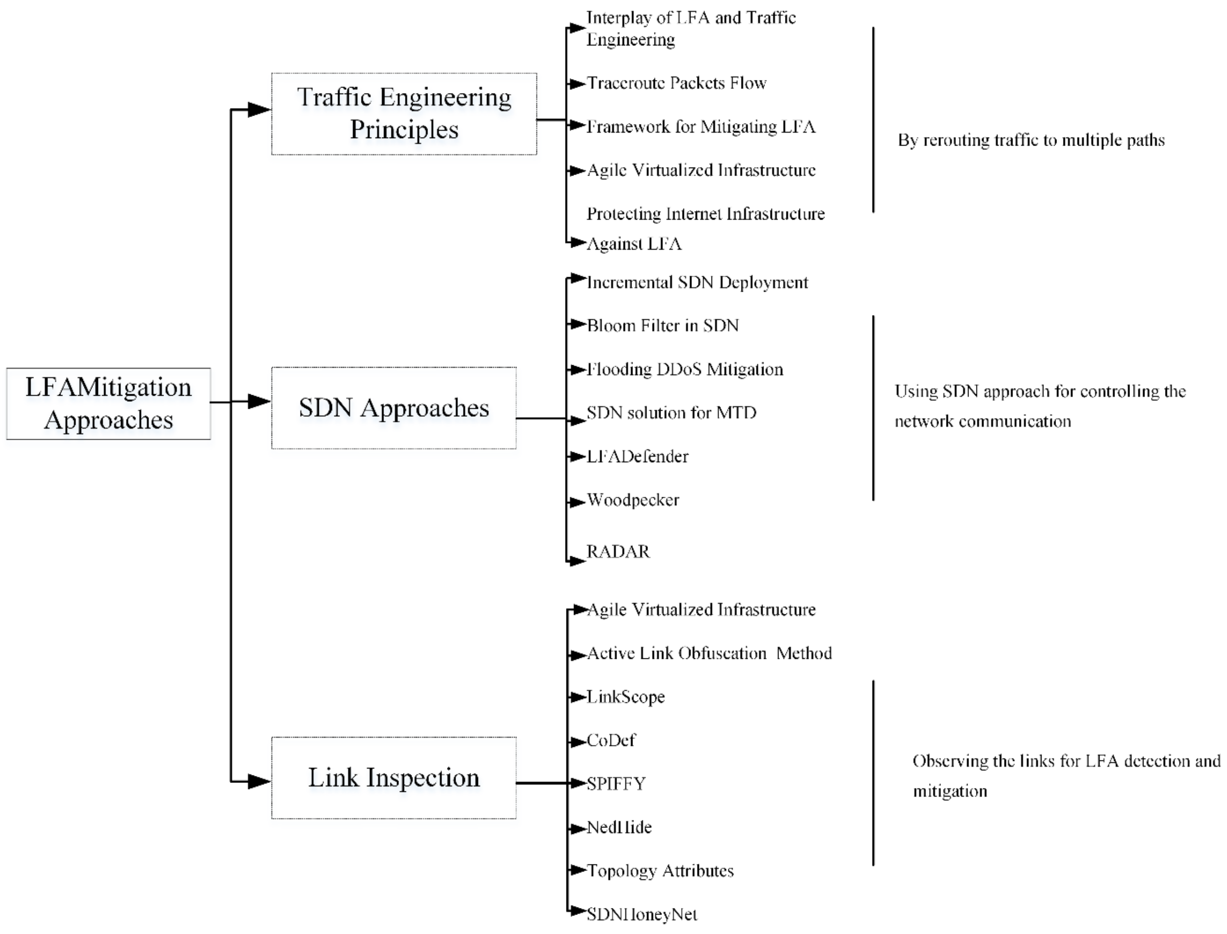
Takayuki et al. [22] propose a proactive strategy to mitigate LFAs that make use of traceroute packets. This technique is based on the fact that traceroute packets are increased in various regions of the network when the network is under LFA. This technique argues that link congestion is a postcursor of the behavior of increase of traceroute packets in a link. In [21] Christos et al. propose a reactive traffic engineering method based on relational algebra principles to mitigate LFAs. This technique is based on the inherent network property of defending against flooding attacks. Whenever the flooding attack takes place, the defender reroutes the traffic, hence, multiple attacker-defender interactions expose the identity of the sources that are consistently participating in flooding events. The sources that change their destination selection to adapt to the re-routing are particularly suspicious. Dimitrios et al. propose a reactive traffic engineering-based method to mitigate LFAs [1]. Rerouting is performed when the defender realizes that there is an attack, the attacker recalculates the traffic path and identifies the critical links again. This work is based on destination-based routing and the variable path which is effective against LFAs.

The limitation of this technique is that the detection speed is widely dependent on the routing rules modification that can cause legitimate traffic delays. Aapo et al. propose a method that combines normal traffic learning, external blacklist information, and elastic capacity invocation in order to provide effective load control, filtering, and service elasticity during an attack [108]. The blacklists come from any IDS or any previous data repository. Fida et al. present an agile virtualized infrastructure method against LFA [54]. This technique employs Virtualize Networks (VN) to dynamically reallocating network resources using VN placement and offers constant VN migration to new resources.

### *2.7.2 SDN Principle-based Approaches*

Wang et al. [25] propose Woodpecker which makes use of incremental SDN deployment to mitigate LFAs. Their technique is based on upgrading routers to SDN switches, which increases the network connectivity. They also use network-probing approach to locating the congested links. In the end, Woodpecker makes use of centralized traffic engineering to balance the traffic across the network and eliminate the bottlenecks that are caused by the adversary during the attack.

Previous techniques do extra header statistics, which increase costs, however, Peng et al. use built-in SDN functionality of flow table inspection [55]. They propose Bloom filter which works in coordination with a collector and detector module. When the utilization ratio of a link is not normal the flow tables are scanned, and abnormal flows are extracted by the parameters of statistical features. The collector module scans flow tables from the SDN network and collects traffic flows by IP header inspection. The detector module extracts IP features from every packet that are important to link attack detection using the Bloom filter.  Abdullah et al propose an SDN based maneuvering technique to defend against LFAs [26]. During the link map construction phase, the links are obfuscated hence, it is difficult for the attacker to launch the attack. Moreover, it is hard to always form the optimal path between links, hence, it increases traffic delays. In [172] a survey has been performed on mitigating flooding attacks using SDN principles. Figure 2-4 a taxonomy of LFA mitigation techniques is provided along with their basic working principles.



LFAMitigation Approaches
Traffic Engineering Principles
Interplay of LFA and Traffic Engineering
Traceroute Packets Flow
Framework for Mitigating LFA
Agile Virtualized Infrastructure
Protecting Internet Infrastructure Against LFA
By rerouting traffic to multiple paths
SDN Approaches
Incremental SDN Deployment
Bloom Filter in SDN
Flooding DDoS Mitigation
SDN solution for MTD
LFADefender
Woodpecker
RADAR
Using SDN approach for controlling the network communication
Link Inspection
Agile Virtualized Infrastructure
Active Link Obfuscation Method
LinkScope
CoDef
SPIFFY
NedHide
Topology Attributes
SDNHoneyNet
Observing the links for LFA detection and mitigation

Figure 2-4 Grouping of LFA mitigation techniques.

### *2.7.3 Link Inspection-baed Approaches*

Qian et al. propose active link obfuscation method which is based on providing fake link map to the adversaries and prohibiting the adversary to accurately analyze the network and create link map to identify potential links [52]. The link map construction phase is one of the most important phases in LFAs, hence, if an adversary is forced to construct a fake link map, then it is challenging for the adversary to locate the targets servers and maintain the attack. The experiments have been performed on the SDN testbed where SVM algorithms are used to distinguish legitimate users from adversaries. The unique flow features of the adversary are extracted from the link map construction phase as well as link flooding phase where SVM is applied to differentiate legitimate users and the adversaries. The limitation of this technique is that SVM is dependent on the volume of the training data where it suffers from low accuracy when the training instances are not enough.

Lei et al. propose LinkScope, which “employs both end-to-end and hop-by-hop network measurement to capture abnormal path performance degradation for detecting LFAs” [16]. LinkScope learns the path metrics of normal traffic, and then classification is performed between the normal and flooding flows. An added advantage of using this technique is that LinkScope can be deployed on one end of the path to perform the measurement instead of installing it on both sides of the link. In this attack, links are carefully chosen where the links with high flow density are selected and bots are used to send low-rate traffic to these servers to congest those links.

Soo et al. propose CoDef which proposes the cooperation among the links that are not attacked by the adversaries during the LFAs to collaborate and legitimate traffic is rerouted to these links for successful network operation [20]. An AS sends reroute requests to all ASs in the network to create a bypass path around the target area to enforce the connectivity in the network.

SPIFFY relies on the principle of temporary bandwidth expansion and rate change measurement to detect adversaries from legitimate traffic [2]. In this technique, the bandwidth of the network is increased for a specific time. Subsequently, a network measurement is performed before and after the bandwidth expansion. The legitimate traffic expands the bandwidth when there is available bandwidth to be used, but the adversaries are not able to increase the bandwidth so, they can be easily detected.

Gkounis et al.[1] propose LFAs in networks and devised a centralized traffic engineering method for limiting the impact of LFAs. The limitation of this technique is that the attack prevention is reactive, it detects the attacks after LFA occurs and reactive measures are taken subsequently where link flooding should have already performed some damage. Wang et al. [52] propose active link obfuscation method against LFAs where a fake link map is presented to the adversaries which causes the adversaries to deviate from the actual target. In this technique, SVMs are also employed for the attack traffic classification however SVM is more accurate when there is a large amount of training data. If the training data is short than the classification process is not accurate enough.

A temporary bandwidth expansion method is utilized where the bandwidth of the network is temporarily increased and in response to bandwidth expansion, the legitimate users will also increase their bandwidth. However, the bots will not be able to cope-up with the bandwidth expansion for long and resulting in consuming all their allocated bandwidth and finally being detected [2]. The limitation of this technique is that if the legitimate users are also not able to increase their bandwidth during bandwidth expansion phase than there will be confusion in differentiating legitimate users and the attackers.

### *2.7.4 Comparative Analysis of LFA Mitigation Techniques*

A huge amount of existing work is present in the field of such attacks on SDNs which are also applicable to traditional networks. However, the mitigation of LFAs in SDN has not yet been explored adequately. We present the crux of the available literature in the field of LFA and provide a meaningful and objective comparison of different LFA mitigation techniques.

Table 2-2 categorizes SDN mitigation techniques based on when the actual mitigation happens. In reactive solutions mitigation is performed after the occurrence of an attack. In a proactive solution, network traffic is handled in a way that the network is always under surveillance which makes it challenging for the adversary to attack. A cost-benefit analysis must be performed before selecting a potential strategy against the LFAs. However, a pro-active solution seems to be a logical choice for mitigating LFAs in SDN.

Table 2-3 summarizes the available literature on LFA mitigation. The first column represents the name of the solution, whereas the second represents a brief summary of the solution. Similarly, column three discusses the limitation of the current technique and the last column represents the year in which the research was published –all sorted in chronological order. It can be easily observed that most of the work on LFA started coming into picture from 2015 onwards. Also, the presented techniques do not cater for the unique features of SDN and are instead applicable to traditional networks only. Some techniques with SDN acronym in there, use SDN’s programmability feature to demonstrate the effectiveness of their techniques originally developed for traditional networks. Figure 2-5 further explains this fact by providing the classification of LFA defense on the basis of network type used. It can be observed that in most of the recent research works SDN-based testbeds are used for evaluation purpose.

This all illustrates the importance of SDN in modern network security research. SDN has been used in demonstrating the effectiveness of various LFA mitigation techniques, however, only a few authors have proposed solutions against LFAs in SDN in general, and none is targeting towards mitigating control channel LFAs.

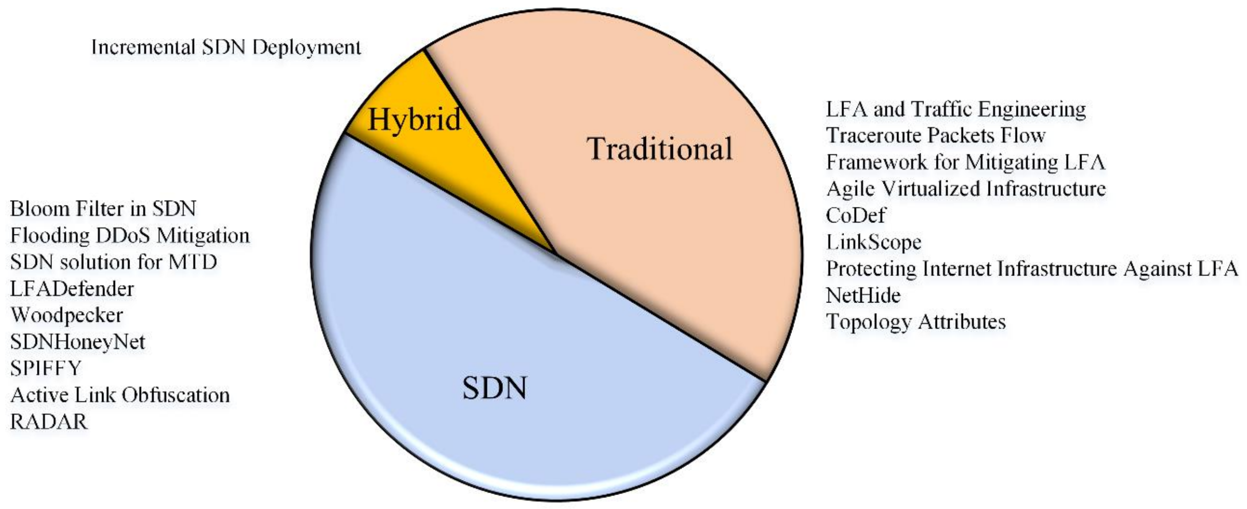


Figure 2-5 Classification of LFA mitigation techniques.

Table 2-2 Classification of LFA solutions based on the response time.

| **Solution Title** | **Solution Type** |
|---|---|
| CoDef [20] | Reactive |
| LinkScope [16] | |
| SDN HoneyNet [19] | |
| Interplay of LFA and Traffic Engineering [1] | |
| Incremental SDN Deployment [25] | |
| Bloom Filter in SDN [55] | |
| SPIFFY [2] | |
| Framework for Mitigating LFA [21] | |
| Flooding DDoS Mitigation [108] | Proactive |
| Active Link Obfuscation Method [18] | |
| Agile Virtualized Infrastructure [54] | |
| Traceroute Packets Flow [22] | |
| SDN approach for Moving Target Defense Attacks [26] | Proactive / Reactive |

Table 2-4 demonstrates the classification of the available literature on type of mitigation technique. Most of the work is around traffic rerouting, and the possibility of LFAs on SDN's control channel is yet unexplored. Traditional techniques can not be used as-is to aid mitigating LFAs in SDN and therefore, modern ways need to be explored in order to address SDN specific challenges, and the benefits available through its unique features such as programmability, ease of use, reconfigurability and extensibility.

After a thorough literature review of LFA in connection to SDN and categorizing the research into groups for objective companions, we have reached to a conclusion that the previous works are based on traffic engineering and link inspection aapproaches and mostly based on a reactive solution.  Moreover, a proactive mitigation solution for LFA resolution on SDN's control channel is yet to be explored.

Table 2-3 Comparative Analysis of Link flood mitigation techniques.

| Solution Title | Main Idea | Limitation | Published Year |
|---|---|---|---|
| CoDef [20] | Autonomous domains that are not affected by bots collaborate during LFA and reroute legitimate traffic in response to requests from congested routers | This solution becomes invalid if all autonomous domains are attacked by the adversary at the same time. | 2013 |
| LinkScope [16] | It is based on learning path metrics of normal flows and on the basis of that detecting abnormal traffic. | Controlling false positive rate is a problem. | 2014 |
| Agile Virtualized Infrastructure [54] | Proactively applying, Virtual networks (VNs) to dynamically reallocate network resources using VN placement. | If the virtual layer is revealed, the adversary can bypass the virtual network to directly flood the target. | 2015 |
| Traceroute Packets Flow [22] | Traceroute packets increase in regions when there occurs an LFA. | It is difficult to classify traceroute commands from legitimate users and attackers. | 2015 |
| Flooding DDoS Mitigation [108] | It integrates normal traffic learning, external blacklist information, and elastic capacity invocation for detection and mitigation. | The source IP address can be spoofed so there can be a partial overlap between the normal user clusters and the attacker clusters. | 2015 |
| Incremental SDN Deployment [25] | LFA detection using centralized traffic engineering based on SDN upgraded nodes. | It detects link flooding after the attack occurs. | 2016 |
| Bloom Filter in SDN [55] | Collector and detector module are used, and flow tables are scanned for abnormal utilization ratio.<br>No extra packet header statistics needs to be performed. | Controlling false positive rate is a problem. | 2016 |
| SPIFFY [2] | Temporarily increases the bandwidth of the network, legitimate users adopt this increase, but bots are unable to increase throughput after expansion, because of depletion of bots and so get detected. | If the legitimate users are unable to increase traffic flow in the temporary bandwidth expansion phase. The false-positive rate is affected | 2016 |
| SDN approach for Moving Target Defense Attacks [26] | Obfuscating the links at attacker link map creation phase using SDN-based maneuvering techniques. | Delay in arriving packets from source to destination, routes are changed, new these paths may not be the optimal paths. | 2016 |
| Framework for Mitigating LFA [21] | Traffic engineering-based solution using attacker-defender interaction. Bots are forced to adopt a suspicious behavior to remain effective, revealing their presence. | Multiple attackers and defender interactions are required to reveal the identity of the attacker.<br>So, it requires an initial set up time for identifying the attackers. | 2016 |

| | | | |
|---|---|---|---|
| Interplay of LFA and Traffic Engineering [1] | Defender module perform rerouting it sniffs an attack, the attacker updates the link map and calculate critical links again. | The detection speed is dependent on the routing rules modification that can cause legitimate traffic delays | 2016 |
| Active Link Obfuscation Method [18] | Linkbait actively mitigates LFA by providing a fake link map to adversaries | It accurately classifies when training data is large. | 2017 |
| SDN HoneyNet [19] | It finds potential links by computing betweenness centrality, bandwidth rate, and minimum interaction set and deploys HoneyNet topology to provide a fake map of the network to adversaries. | Accurately identifying the bottlenecks is difficult, so weights need to be associated with the parameters to assess. | 2017 |
| Protecting Internet Against LFA [17] | ISP cooperation and traffic rerouting by incentivized routing strategy where links with low transmission are bypassed. | This strategy works regardless of LFA occurs or not, which creates overhead on the network by rerouting traffic to other routes increasing the source to destination time. | 2018 |

Table 2-4 Categorization of LFA mitigation research based on different features.

| # | Research Paper Title | LFA Mitigation Technique | Category |
|---|---|---|---|
| 1. | Incremental SDN Deployment | “Updating M nodes to maximize the network connectivity by monitoring the flow rules updating” | Link Utilization |
| 2. | Active Link Obfuscation | “By providing the faked link map to adversaries, Linkbait can actively mitigate LFA Linkbait intends to hide target links and use bait inks to fake target links.” | Link Inspection |
| 3. | A Novel Framework for Modeling and Mitigating LFA | “After the flood is known, the defender balances the load by re-routing traffic destined to different destinations. Sources that change their destination selection to adapt to re-routing are particularly suspicious” | Traffic Rerouting |
| 4. | Interplay of LFA and Traffic Engineering | “The attacker monitors the network routes and reacts to routing changes performed by the defender. Bots will then change their decoy server selection in case the re-routing has diverted their load from the critical link(s), repeating the cycle, thus revealing their identity” | Traffic Rerouting |
| 5. | CoDef | Providing bypass links by AS rerouting around flood area. “An adversary has to make an invalid choice: Conform to rerouting and give up attack persistence at the targeted link Not conform and have its flows discovered and bandwidth-limited.” | Traffic Rerouting |
| 6. | Towards Defeating the Crossfire Attack using SDN | Balances traffic load by rerouting traffic destined to different destinations, without knowing the attacker’s classification. Records sources observed in DoS’ed links to detect suspicious recurring sources. | Traffic Rerouting |
| 7. | SPIFFY | Increasing the bandwidth of the link and then taking back to the original. Bandwidth before and after the expansion period is calculated bots will not be able to maintain the bandwidth | Bandwidth Expansion |
| 8. | SDN HoneyNet | Provide a fake link to the attacker by employing graph metrics statistics | Fake link |
| 9. | LinkScope | “Reactive solution based on learning path metrics of normal and detecting abnormal traffic” | Link Inspection |
| 10. | Agile Virtualized Infrastructure | Proactively applying, “Virtual networks (VNs) to dynamically reallocate network resources using VN placement.” | VN Placement. |
| 11. | Traceroute Packets Flow | “A number of traceroute packets increases in regions when there occurs an LFA.” | Traffic Rerouting |
| 12. | Protecting Internet Against LFA | Cooperation among sources and destination autonomous domains | Traffic engineering |

Table 2-5  Taxonomy of LFA mitigation techniques based on performance metrics.

| **Technique** | **Accuracy** | **Detection Time** | **Solution Type** | **Approach** | **Complexity** | **Method** | **Scalability** | **Evaluation** |
|---|---|---|---|---|---|---|---|---|
| CoDef [20] | High | Low | Reactive | Traditional | High | Traceroute | High | Simulation |
| LinkScope [16] | High | Low | Reactive | Link Inspection | Low | Link Inspection | High | Real testbed |
| Agile Virtualized [54] | High | Medium | Proactive | Traditional | Low | VN placement | High | Real testbed |
| Traceroute Packets Flow [22] | Low | Low | Proactive | Traceroute | Low | Traceroute | Low | Simulation |
| Flooding DDoS mitigation  [108] | Medium | High | Proactive | ML | High | SDN-based | High | Real testbed |
| Incremental SDN Deployment [25] | Low | High | Reactive | Woodpecker | High | SDN-based | Low | Simulation |
| Bloom Filter in SDN [55] | Low | High | Reactive | Bloom filter | High | SDN-based | Low | Simulation |
| SPIFFY [2] | Medium | High | Reactive | Traditional | High | Link Inspection | Medium | Simulation |
| SDN Approach for Moving Target Defense [26] | Low | High | Proactive/Reactive | SDN-based | Medium | SDN-based | Low | Simulation |
| Framework for Mitigating LFA [21] | Medium | High | Reactive | Traditional | low | Traffic Rerouting | Medium | Simulation |
| The interplay of LFA and Traditional [1] | Medium | High | Reactive | Traditional | Medium | Traffic Rerouting | Medium | Simulation |
| Active Link Obfuscation [18] | High | Low | Proactive | Link obfuscation | Low | Link inspection | High | Testbed/Simulations |
| SDN HoneyNet [19] | Low | High | Reactive | HoneyNet | Low | Link Inspection | Low | Simulation |
| Protecting Internet Infrastructure Against LFA [17] | High | Medium | Proactive | Traditional | Low | Traffic Engineering | High | Real testbed |

## 2.8 Flood Mitigation techniques

In this section, we discuss the link flood mitigation approaches which can be used to mitigate the LFA. There are mainly three approaches to mitigate flood namely Sinkholing, Scrubbing, and Null Routing. In the interest of space and to keep the reader focused only very brief info Sinkholing and Scrubbing will be provided, as the detailed literature can be found at [173, 174, 236].

### *2.8.1 Sinkholing*

Sinkholing manipulates the data flow in a network it has been used to redirect traffic from its original destination towards a host/server of choosing. This technique can also be used by the adversaries to divert legitimate traffic from the intended destinations. However, the security operators use this technique to divert the attack traffic towards the chosen destinations to react to attacks. A simple example to explain Sinkholing is when we type a web address on the browser, it takes you to the destination website where the DNS is responsible for taking care of the destination IP. However, if the domain address is sinkholed then the browser will take you to the sinkholed address. Sinkholes rely mostly on the DNS to forward the traffic to a carefully chosen sinkhole. This process involves the monitoring of the domain name where the network security operators usually try to setup automated mechanisms to control a malicious domain name at the time of the expiry of the registry. A customized sinkhole can be created that has the ability to forward the traffic from the original IP address to a sinkhole address using router modification or firewall settings. Sinkholes are widely being used by the security operators for day-to-day surveillance activities.

As botnet IP addresses may also be used by real users, Sinkholing is prone to false positives.

### *2.8.2 Scrubbing*

This technique is "an improvement on the arbitrary Sinkholing technique, scrubbing routes all ingress traffic through a security service. Flooding packets are identified based on their header content, size, type, point of origin, etc. The challenge is to perform scrubbing at line rate without causing lag or otherwise impacting legitimate users" [236]. Figure 2-10 illustrates the underlying mechanism of the scrubbing technique for flood attack mitigating.

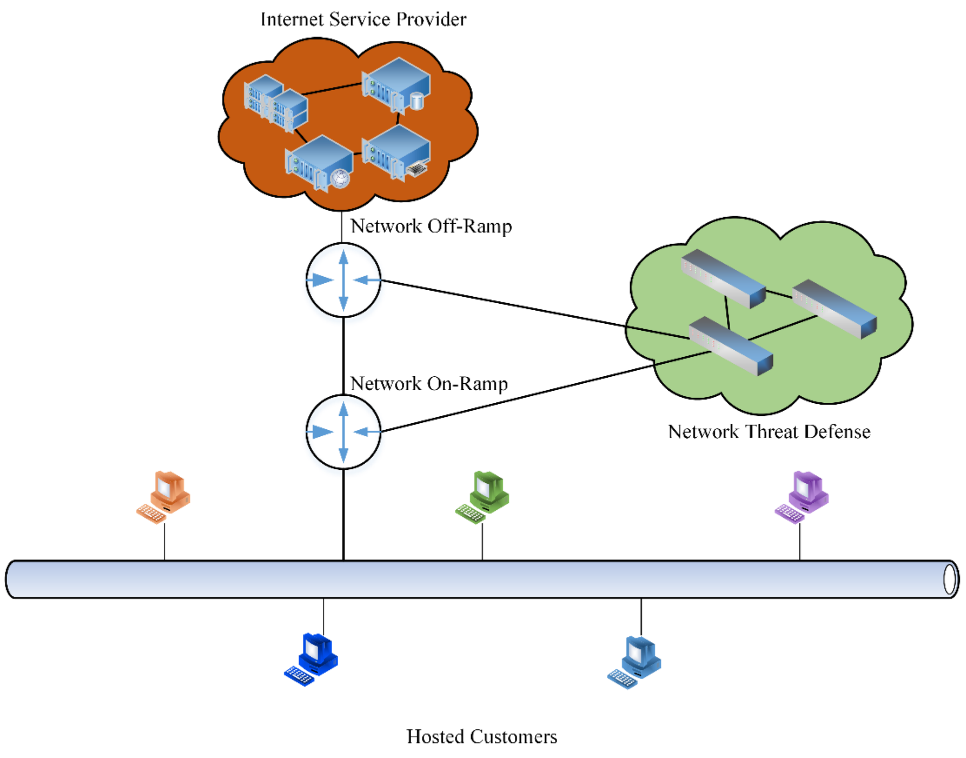


Figure 2-6 The mechanism of scrubbing the flood traffic.

Scrubbing is understood as an important unit to conduct traffic analysis, cleanse data and remove certain traffic. Scrubbing techniques are mainly used by cloud service providers or ISPs which tries to off-ramp the data for cleansing purposes. The traffic is redirected towards the scrubbing centers where the attack mitigation system eliminates the attack and forwards the attack traffic back to the data centers.

The scrubbing centers should be equipped with handling high volumes of flood at the application and the network layer. Moreover, it also involves more human intervention and blocks more good traffic instead of the attack traffic. An alternative to Scrubbing is Null routing.

### *2.8.3 Null routing*

NULL routing or blackholing sends selected traffic to an invalid address. On defining a valid route, network then starts sending traffic to specific IP addresses through it. However, a NULL route tells the system to drop the traffic destined to a specific IP. This implies that any TCP server will not be able to send an SYN/ACK reply to the sender. And UDP traffic will be received, however, no reply would be generated to the requesting IP.

NULL routing could cause a higher rate of false-positives by disposing-off some legitimate packets as well.

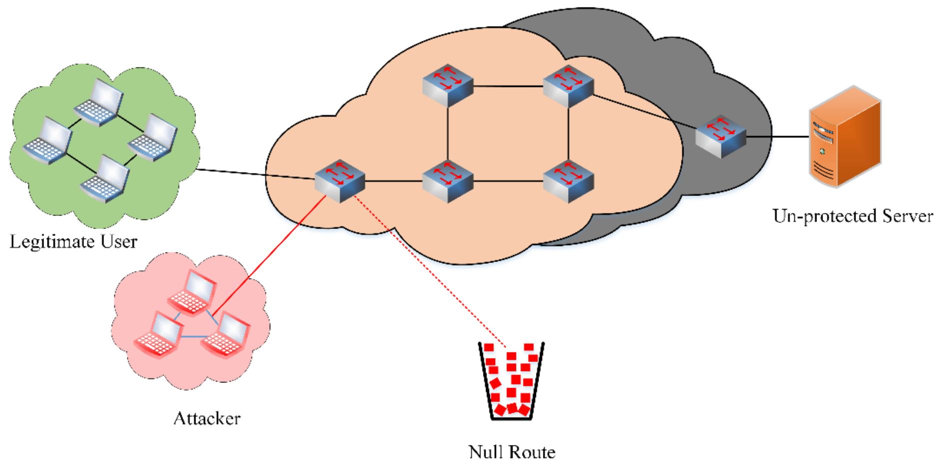


Figure 2-7 The working principle of NULL routing.

Figure 2-7 describes the working principle of the NULL routing. It can be assumed as an IP address which does not have a destination. Hence, any traffic sent to that IP address will be discarded. It is an efficient technique as when the data moves among the routers, they need to have the destination address towards which they route the traffic. This is performed by broadcasting the status among the network routers about which IP address they are processing and where should they route the packets that are aimed towards them.

Whenever an IP address has been NULL routed then it is broadcasted as having no destination in the network so any data forwarded to this address will not be able to make up to the destination and will be discarded. Hence, the network will not have to process this data and will be safeguarded from the flood traffic. It is an essential technique that is used for flooding attack defense that has no other ways to block the malicious flows. When the flooding attack occurs, the target infrastructure may not be the only entity that will be affected by the attack, other hosts using the same infrastructure will also face the impact of the attack. Where this attack will result in a high risk to the server and the indirect host. The defense mechanisms insert a NULL route of the victim into the network and start blocking the malicious flows to safeguard the network. However, this technique brings a drawback where it alleviates all the malicious traffic but also eliminates the good traffic too. Hence, this technique is catastrophic for the users whose business relies on the 24/7 availability of the internet. If the defense mechanism routes all the traffic to the NULL route, then it badly impacts the legitimate traffic. Moreover, most of the flooding attacks are spoofed hence, applying NULL routing on the source is almost impossible.

Furthermore, NULL routing sometimes becomes more damaging for example an ISP with the residential customers need to NULL route into their infrastructure which invokes service disruption of the hundreds of legitimate customers. NULL routes are usually configured as using a special route flag; however, they can also be used by routing the traffic to an illegal address such as 0.0.0.0.

NULL routing is easy to implement than the traditional firewalls since it is available on the routers moreover it does not pose high overhead on the network performance. Due to the availability of high-speed routers, NULL routing can sustain higher throughput than the traditional firewalls. This is the reason why NULL routing can be used in high-performance vital routers to block large-scale flooding attacks before reaching the destination. However, this technique is also vulnerable to the specially crafted packets that avoid NULL routing security.

#### 2.8.3.1 Creating a NULL route

NULL routes are created through ‘route’ command in windows and UNIX operating systems.

The following set of commands demonstrate creation and usage of NULL routes.

For example, if we are getting unwanted SSH requests from an IP `198.168.0.95.` Then:

```
#route add 193.252.19.0.0.0.0.0
Add net 193.252.19.0: gateway 0.0.0.0

route@server:~#netstat -na | grep: 22
tcp 0 0 0.0.0.0:22 0.0.0.0:*LISTEN
tcp 0 0 192 168.0.197:22 192.168.0.195:57776 ESTABLISHED
```

IP command is used to add a NULL route

```
 route@server:~# ip route add blackhole 192.188.0.195/32
```

`IP route shows` command can be used to check whether the route is in place

```
route@server:~# IP route shows
default via 192.168.0.1 dev eth0 metric 100
blackhole 192.168.0.195
```

Subsequently, the connections established will act as time out moreover, the subsequent connections from the blocked IP will receive the following.

```
baduser@attacker:-~$ ssh 192.168.0.197
ssh: connect to host 192.168.0.197 port 22: No route to host
```

The NULL routing is also often called as the blackhole routing. All the current routers are equipped with handling such traffic. Like in Cisco's terminology NULL route is called the NULL interface which is used to create a black hole. Static routes are created for the destinations whereas the static route configuration points to the NULL interface.

After a thorough comparison of various flood mitigation techniques presented above, based on the characteristics of each we shortlist NULL routing as a useful technique to help eliminate control channel LFAs in SDN.

## 2.9 Classification of LFA Mitigation Techniques

The objective of this section is to come up with the qualitative performance features of LFA mitigation techniques. The aim is to introduce these features and quality metrics as a must-have in the solution that we are going to propose in the upcoming chapters.

A thorough study of LFA mitigation techniques has demonstrated that researchers have examined this problem from multiple angles. We did not find a single solution that addresses all the challenges posed by this attack, with some aspects have been more thoroughly studied than the others. In the absence of an objective criterion, it is hard to judge the quality of a suitable solution. In this regard, we have proposed a set of quality metrics for systematically examining the strengths and weaknesses of LFA mitigation techniques based on an objective criterion that uses a set of features to rank and quantify them. The qualitative performance features are discussed below:

### *2.9.1 Detection Accuracy and Detection Time*

The accuracy with which an LFA mitigation technique alleviate the flood traffic where false-positive rate is low. Higher accuracy is desirable in LFA mitigation strategies. We categories the LFA mitigation techniques according to their experimentation evaluation to detection accuracy levels of high, medium, and low.

Time that the proposed solution takes to detect LFA after it occurs is defined as the detection time. It largely depends on the underlying technology used in the LFA defense. Some techniques are based on the analysis of the rise in the traffic flows in a specific region as a detection method against LFAs. To carry out such type of detection, the defense mechanism should have to analyze the whole network for attack traffic analysis, however, high traffic rate can also be caused by the legitimate traffic. So, the defender needs to distinguish between the legitimate and the flood traffic. Hence, detection may yield more time. Similarly, some LFA defense techniques utilize ML classification strategies which may yield comparatively lesser time.

Detection time plays a significant role in the acceptability of a defense solution. The research in the LFA detection and defense is in its early stage hence only limited defense mechanisms are available. There is still a need for an optimal solution for LFA defense. Low detection time is desirable in LFA mitigation techniques as a solution taking more time in order to respond to LFA, the adversaries would have already done damage to the network. The detection time is also classified into high, medium and low, where time-efficient techniques will be categorized as low. There is another aspect of detection time where proactive solutions try to establish strong surveillance of the network before the occurrence of LFA. Therefore, proactive solutions have low detection time.

### *2.9.2 Solution Type And Approach Used*

There are two types of mitigation strategies, i.e. proactive and reactive. The proactive strategies setup a prior defense breach in order to avoid LFA before it occurs. These techniques are based on zero trust security where no connection is trusted unless it has been deliberately allowed. The proactive solutions comprise up of diverse setup tools, for example, network visibility, monitoring, and segmentation at various network levels. Alternatively, the reactive strategies perform the alleviation after the threat has penetrated the network and then react using the pre-defined set of solutions. Most of the prevailing defense solutions are reactive.

The best defense against LFA is proactive solutions, where precautionary measures are taken prior to the attack. The reactive solution operates subsequent to the attack occurrence which makes the resources of the network vulnerable of security risks. In the reactive methods, the information at the edge routers and switches is collected and meta-data

analysis is performed to detect the attack. If the detection finds the malicious packet or attacks it provokes the mitigation procedure. A reactive solution uses the flows that are already in the network hence, these solutions are more cost-effective.
A proactive method uses an inline tool to have a complete visibility of the network traffic. This mechanism analyzes every packet it received and uses the underlying predetermined patterns to compare the traffic.  Based on the definition it is always better to devise a proactive solution as it is always active. Proactive solutions are mostly used where there are real-time needs for example video games, video streaming, voice, and medical, real-time air control communication and other critical infrastructure. Proactive solutions are always active hence they cost more and induce a certain overhead. In proactive solutions, network traffic is handled in a way that it is very difficult for the adversary to launch an attack. Therefore, we find it as the best way to mitigate LFA in this way the damage incurred by LFA is minimum. However, the overhead attached of using these solutions is the main consideration for the selection. The decision to use the type of solution depends on business needs. If the business can afford the higher cost and need precise security then the proactive solution is a better choice. Alternatively, if the business can afford some delay in detection and do not have real-time requirements then a reactive solution can work fine.

The ‘Approach Used’ corresponds to the type of approach used in the experiment, we categorize these techniques as traditional, link obfuscation, and ML. Traditional techniques correspond to the approaches widely being used for network operation like packet inspection and flow filtering. ML techniques correspond to using ML approaches for detection and mitigation of LFA like incoming traffic classification to identify malicious flows and adversaries. Link inspection techniques correspond to constantly observing the links to identify the malicious activity over the network.
Traceroute techniques use traceroute packets in order to alleviate and detect an attack. These techniques analyze traceroute packets increase phenomena to identify the malicious activity as adversaries send multiple traceroute packets to create a link map of the network before launching the attack. For attack mitigation, these techniques reroute the attack traffic so that it is unable to reach the destination. In the same way, link obfuscation techniques, deceive the adversary to create a correct link map and identify the potential target by

obfuscating the network links. HoneyNet techniques provide virtual connectivity over the nodes attacked by the adversary, this way traffic can be bypassed from the attack point. The Bloom filtering approaches use a probabilistic data structure to detect LFA. In the same way, the Woodpecker technique incrementally deploys SDN strategies in order to enable centralized control of the network and alleviate LFA.

### *2.9.3 Solution Complexity, Analysis Method and Scalability*

Although security is one of the prime concerns in the current network paradigms, multiple defense mechanisms are used to perform the security of the systems. However, this defense mechanism put extra overhead over the network by employing messaging or other computation techniques to secure the network. Moreover, client-server messaging during security also puts extra overhead on the network. More messages consume greater network resources.

Complexity is the amount of resources that will be consumed by the solution in order to mitigate the attack. These resources may correspond to time to detect or mitigate an attack or other resource requirements consumed by the solution in order to complete its operation. Lower complexity values are preferred for an optimal solution against LFA. Here we classify the techniques into low, medium, and high complexity solutions by analyzing the experimental setup provided by each technique.

The ‘Analysis Method’ corresponds to the type of solution that has been used by the underlying approach. The examples of analysis methods are traffic engineering, traceroute, and SDN-based approaches.

Traffic engineering is a technique to optimize network performance by dynamically predicting, analyzing, and regulating the data transmitted over the network. It can be employed on all types of networks. It facilitates efficient network operation and concurrently optimizing the performance of the network. This technique moves the traffic flood away from the congested areas to comparatively less flooded links. When the level of traffic on the network reaches to an allocated limit, the network is denoted as congested. In this condition, the traffic engineering principles come into play to alleviate the traffic flood on certain links and deviate the traffic to other paths.

The traceroute techniques are employed to display the route and identifying the transit delays of packets across the IP network. A history of the route is recorded and analyzed by measuring the Round-Trip Time (RTT) of the packets received from every successive host on the route. Moreover, the sum of the mean times in each hop is a measure of total time spent in devising a connection. The process of traceroute continues until all the sent packets are lost more than twice, this way the connection is lost and the route cannot be evaluated. The difference between the approach used and the analysis method is that the approach quantifies specific categories of the technique used against LFA. However, the analysis method categorizes the solution into broader categories of the techniques.

Scalability is the property of the LFA mitigation system to accommodate the growing amount of task by adding more resources. A routing protocol is considered as scalable according to the network size if each node grows as *O (log N)* where N corresponds to the number of nodes in the network. The capability of an LFA solution to be deployed on large scale networks is called the scalability. This metric must be considered extensively before devising an LFA solution. As more businesses are getting online, the need for a scalable solution is highly inevitable.

Moreover, the prevailing solutions need hardware to perform link-level measurements which may require additional hardware in case the network grows. This phenomenon play a negative impact on the scalability of the system and can result in the crashing of the security solution. Without scalability planning, the costs grow, and efficiency comes down. The basis behind planning the scalability is analyzing the security solution development in a way that it can be expanded in all the ways including the underlying network size, the data rate, and the attack severity.

The concept of scalability can be understood easily however, it becomes difficult to implement in the current paradigm of heterogeneous infrastructure. We categories the scalability of solutions as, high, medium, and low. Current networks are expanding continuously, so a feasible solution must possess the quality of being highly scalable. Most of the available research is based on simulation results, which cannot be tested for scalability.

### *2.9.4 Evaluation Method Used*

Since the computer networks have currently become too complex for traditional experimentation methods to predict an accurate understanding of the behavior of the system so it has become a challenging task to perform experiments using traditional networks. In computer security research, the network simulation provides a mechanism whereby a software program can model the behavior of the attack and its defense. The infrastructure entities like routers, switches, nodes, access points, and links can be created through the programming model. Most of the network simulators use discrete event simulation where the variables tend to change in discrete time stamps.

And the user can alter the behavior of the components according to the required configurations. Multiple attributes can be modified in a logical manner and experiments can be performed on how they behave under different conditions. Current network simulators are nowadays equipped with latest communication technologies like LAN, Wide Area Networks (WAN), LTE, 5G, Bluetooth, Wireless Sensor Networks (WSNs), and IoT. The network simulators come with a variety of options like GUI-based and CLI-based simulators. The network model illustrates the nodes, links, switches, and events. The output metrics may include data rate, packet loss rate, bandwidth consumption, etc. The log files contain information regarding every packet and every event occurred during the simulation.

Moreover, to perform experiments in a software environment that the components used mimics the actual hardware, the network emulators are employed. The methodology is that the real packets are transferred using a live application to an emulation server, then those packets get modulated in a simulation packet. Finally, the real packets get demodulated after experiencing the effects of delay, packet loss, and jitter, etc. which injects these effects to the rea packets in a way that the real packets have been suggested to real network conditions.

During the past few years, network technology has been progressed in a rapid manner. The motive was to reduce the cost and increase the efficiency of the current networks. Network security research involves real-time experiments to analyze the impact of a solution in a physical system before employing it. The physical testbed is mostly hosted by large organizations which can be hired by the researchers to perform the real-time experiments.

The physical testbeds include real hardware specially designated for performing real-time experiments. Figure 2-8 illustrates the classification metrics of LFA mitigation techniques.

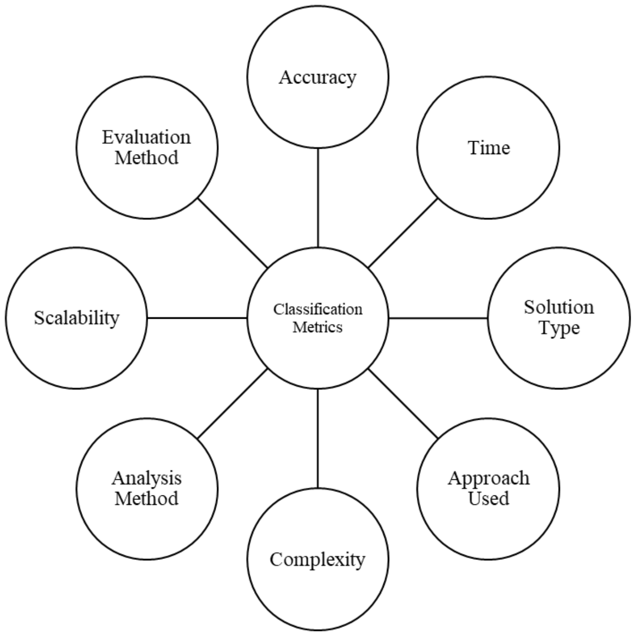


Figure 2-8 LFA Classification metrics.

Different methods are used to perform the evaluation of the proposed solution. Some authors use simulations to demonstrate their concept while others use real testbeds. Reference is given to solutions that are tested on real testbeds.

In the Table 2-5 we have compared LFA mitigation techniques based on above-mentioned metrics. CoDef and Linkscope have high accuracy however they work in a reactive manner that makes them vulnerable to the security issues. Alternatively, the agile virtualized technique possesses the qualities for a good solution as it has higher accuracy in detecting the malicious flows as well as it is highly scalable. It also works in a proactive manner and moreover, they have tested their solution on a real testbed. SDN HoneyNet is a novel solution however, it has low detection accuracy and it can suffer from the problem of more time to deploy the HoneyNet topology because it computes different graph-based measurements to identify the vulnerable nodes. Another drawback of this technique is that it is not tested in the real testbed scenarios. SDN approach for moving target defense provides a solution that works fine in both proactive and reactive sceneries. In the same way, it takes a lot of time in order to detect LFA. This solution has not been tested on the real testbed and it also lacks in qualifying scalability requirements.

The selection for a reliable solution against LFA depends on many factors as provided in the comparison metrics. It depends on the requirements of the environment where a solution needs to be deployed. Because no solution is perfect in terms of all the defined metrics. In some situations, we require the highest accuracy, so we can compromise other quality metrics in order to attain accuracy requirements.

Some basic insights for choosing a mitigation strategy is that the solution should have a low false positive rate and low attack detection time. The solution must also be scalable so that it can be deployed on large scale networks. As mentioned previously that there is a certain tradeoff while selecting an optimal strategy against LFA.

The fact that LFA uses low-rate traffic to attack potential targets, makes it difficult to mitigate. However, a practical mitigation strategy would be the one that incorporates the source and destination AS coordination against LFA. Similarly, the solution must be tested on a real testbed while fulfilling the scalability requirements. In our view the real solution to mitigate LFAs against control channel attacks, should work at line speed and would ideally rely on machine learning to intelligently detect flooding situation and eliminate it through NULL routing before it harms the network.

## 2.10 Chapter Summary

This chapter provides a holistic view of classifying the existing works available in the domain of LFA defense. We provide different classification metrics to categorize the literature. A detailed taxonomy of LFA mitigation techniques with the related performance metrics has also been presented. This comparative analysis helped us answer a few research questions and demonstrated that the adequate attention has not been given to mitigating LFAs on SDN, and that the control channel has yet to explored for attacks and vulnerabilities. Through this study it became clear that due to its simplified configs, SDN setups are being used to conduct research for traditional networks. Therefore, quite a few papers do talk about study of LFA demonstrated on SDN testbeds, but the mitigation techniques are not necessarily targeted for SDNs and their unique operational nature. The control channel in SDN is a critical resource and if this link is flooded, the whole network can malfunction.

In this chapter, we bring forward all the variants of SDN to help study if the identified weak points are valid for all the types. On finding out control channel as a weak link that could be explored for LFAs in all SDN types, we then identify LFA as a lethal attack against SDN and for its variants. We classified the prevailing LFA mitigation techniques into different categories based on underlying detection and mitigation mechanisms. We then reached to a conclusion that there was a lack of defense mechanisms on detecting and mitigating LFA in SDN, specifically on the control channel that can easily become a bottleneck.

Through this chapter we were able to select a general technique to mitigate floods on any link –NULL routing. We also short listed the quality metrics and features that a practical solution must have. And based on the characteristics we reached to a conclusion that an intelligent mechanism is required to detect and mitigate floods at line speed.

In the next chapter, we will formally present the problem statement and provide the architecture of our proposed solution named as CyberPulse. We will discuss different modules of CyberPulse and elaborate on the working principle of the solution in detail.

# CHAPTER 3. PROBLEM DEFINITION

SDN has proved itself to be the backbone for the current data center networks and is currently becoming an industry standard. The big data and virtualization need of the current networks greatly stress that we use SDN for network provision and management. With the greater dependency on the communication networks, any discrepancy can lead to disruption of the network which may cause inefficiencies in the network management and even disruption of the services. This can cause severe data and investment loss. In this context, LFA can cut off specific areas of the networks from the others using low-rate legitimate traffic. It poses a great security threat to the modern networks and can have a devastating impact on SDN which can bring individual planes down, disrupt the communication between the planes, and ultimately bring down the whole network.

However, the sensitive nature of the control channel can make this a single point of failure for the whole network. So, network community should give serious attention to successfully implement and upgrade SDN architecture against LFAs. Therefore, mitigating LFA on the control channel of SDN poses a vital challenge. In order to deal with this threat, we propose an LFA mitigation framework on the control channel of the SDN named as CyberPulse.

The previous chapter explains that there are profound vulnerabilities of LFA on the control channel. There is also a lack of literature available that addresses these weaknesses that demonstrates the vital importance of this study. In this chapter, we will discuss how LFA can be mitigated using CyberPulse security framework. We discuss the problem of LFA in SDN and elaborate on how LFA can severely impact the control channel of SDN. Subsequently, we will discuss the proposed CyberPulse solution against LFA. We elaborate on different components of CyberPulse including the Link Listener, Flood Detection, Flood Mitigation, and Deep Content Inspection (DCI) module.

We explain design considerations for the framework against LFA in SDN and demonstrate how LFA detection can effectively utilize pattern matching technology in SDN. Hence, we provide a detailed study of the pattern matching technology and its effectiveness in detecting anomalous traffic in the network. Subsequently, we evaluate the pattern matching technology using FA mechanism and provide detailed examples to understand the packet inspection techniques employed in CyberPulse. We performed extensive experiments to compare the complexity of the compilation and matching phase of DFA-based RE processing engines which includes RE2, Rexgrep, and Regex-Processor.

## 3.1 Problem Statement

The separation of control and data plane in SDN provides a higher level of programmability and provoke a more flexible way to manage and control the networks. However, the limited bandwidth between the control and data plane can become a bottleneck in the communication where SDN may undergo higher packet loss and delay when the control channel link is under attack. The previous solutions only rely on one aspect of the problem where, however, hybrid solutions are needed in a current versatile network environment which has the ability to flexibly adapt to the user requirements. The attacker can aim at realizing this bottleneck and launching attacks on this plane. The controller in SDN mediates the whole network traffic by installing flow rules on the OF switches. Figure 3-1 illustrates the step two-layer security against LFA detection and mitigation.

LFA is considered lethal against modern SDNs because of three reasons, first, it uses legitimate IP traffic which makes it harder to filter out rather than using spoofed IP addresses which can easily be distinguished. Second, it sends legitimate packets to publicly accessible servers called the decoy servers. Hence, a coordinated attack keeps these packets flowing without any interruption causing a flood on the attacked links. Third, it transmits low bandwidth flows from each bot individually, these legitimate flows then cumulatively flood certain links in the network without being detected. The controller of SDN accompanies vulnerabilities for LFA as it sends control traffic which is the most critical information for SDN as it performs all the crucial tasks of flow rules installation, network configuration, and optimum path calculation.

In LFA, the prime concern of the attacker is to disconnect the controller path links to make it ineffective. LFA poses a great security threat on the SDNs, especially the centralized control mechanism can be disrupted using SDN. Since LFAs use low-rate traffic which resembles legitimate traffic, it cannot be detected using the traditional detection and defense mechanisms. Moreover, the centralized network management in SDN portrays that only the controller has access to the data plane switches which stresses the need for solutions that can leverage the SDN characteristics of centralized management. In this context, we propose CyberPulse to effectively mitigate LFA on SDN control channel. CyberPulse utilizes the centralized network management of SDN to effectively collect network statistics and classify them into benign and flooding flows.

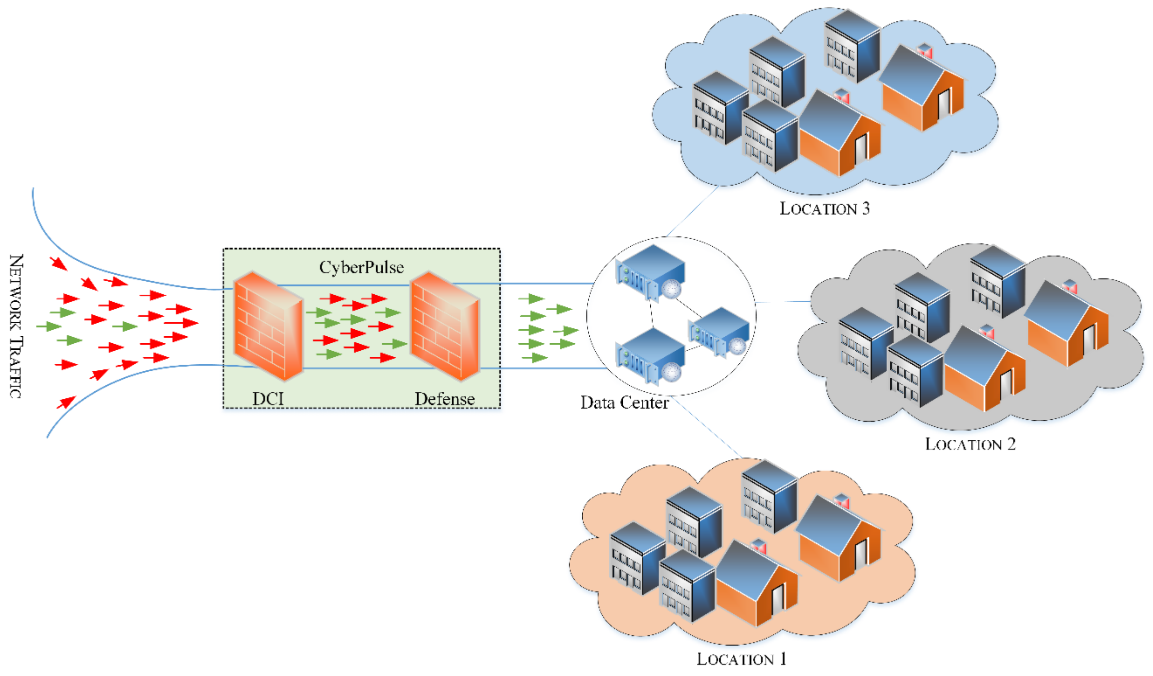


Figure 3-1 An overview of the proposed security solution. The network traffic has been subjected to traffic monitoring solutions, where the final network traffic to the destination has been filtered out against the LFA.

In the same way. The data plane is the main working unit which is dependent on the controller communication, in SDN where it derives all the information through the hardware devices, attacking data plane devices result in disruption of the services of the whole network. Similarly, the control channel is the most critical resource because all the control information passes through this link, as this is a single link and any congestion can cause a delay in information.

The adversaries can take hold of this link by implying hard to detect LFA and congest the link in a way that it can become irresponsive resulting in disconnection of the whole network. Control channel has the tendency to be flooded by a bunch of traffic hence, there must be a link-scanning mechanism that can check whether the congestion is caused by the legitimate traffic or it was caused by an attack. Since the control channel doesn't have any detection and mitigation strategy, there is a strong need to imply link congestion detection and mitigation mechanisms.

So, we provide a solution called CyberPulse for the problem of LFA on the control channel of SDN by employing ML and JSON-based DCI techniques [175]. We discuss the main components of CyberPulse including its modules and a detailed evaluation of JSON-based DCI in this chapter.

## 3.2 Overview of LFA Defense Components

We discuss CyberPulse framework including its components, the configuration of its components and a detailed working principle of the framework. CyberPulse security framework has been presented in Figure 3-2, we can observe that there are three main components of our proposed framework i.e. Link Listener Module, Flood Detection, and Flood Mitigation Modules. There are two planes connected by a link i.e. the control plane and the data plane. CyberPulse continuously observes the control channel and perform the mitigation operation in case of any congestion found on the link. The Link Listener implements link listening algorithm which also includes a submodule called DCI. The link listening algorithm continuously listens to the traffic on the control channel and collects statistics of the network traffic.

Link Listening Module sends the network statistics to the Flood Detection Module. These statistics are utilized by the Flood Detection Module to separate the benign traffic from the flood traffic on the basis of ML techniques. This ML technique is trained by using the training dataset. These three modules collaborate with each other in order to perform the flood mitigation process and keep the network clean from flooding.

The module which interacts with the control channel directly is the Link Listener Module. With the help of the link listener algorithm, it performs the surveillance process on the link. It continuously observes the link for congestion alert, and if it finds any congestion it invokes Flood Detection Module which performs the analysis on the type of congestion on the link i.e. flood attack or normal traffic congestion. If it finds normal congestion than it allows the link to perform its operation but if it finds any flood traffic, it invokes flood mitigation module. The Flood Detection Module performs the analysis on the acquired network statistics and separates the legitimate traffic from flooding flows. It is responsible to identify the sources that are involved in the flooding of the link. Flood Detection Module passes on the identified flood traffic to the flood mitigation module that eliminates the flooding flows and enable smooth operation of the network.

LFA mitigation framework deploys ML technique to classify the network traffic into benign and malicious categories. In this context, we have identified different DoS flood attack mitigation studies, the name of the ML technique, and the features used for the traffic classification in Table 3-1.

Table 3-1ML techniques in flood attacks mitigation.

| Technique Name | ML Algorithm | Features Used |
|---|---|---|
| Distributed-SOM [176] | Distributed Self Organizing Maps | Number of flows |
| | | Number of packets per-flow |
| | | Number of bytes per flow |
| | | Time duration |
| Adaptive Artificial Immune Networks [96] | Artificial Immune System | Traffic analysis |
| DyProSD [177] | C4.5<br>Naïve Bayes<br>Decision Tree | Source IP |
| | | Destination IP |
| | | Sampled Interval  Time |
| | | Flow ID |
| | | Total number of connections |
| OpenFlowSIA  [178] | SVM | Protocol |
| | | Source IP |
| | | Source port |
| | | Destination IP |
| | | Destination port |
| DDoS Flooding attack Detection Algorithm  [179] | Hop Count Filter Algorithm | Source IP |
| | | Destination IP |
| TDFA [180] | IP Trace-back algorithm | Traffic analysis |

LFA mitigation framework helps in securing the control plane communication with the data plane. It is pertinent to note that none of the aforementioned strategies solve the LFA problem on the SDN’s control plane to data plane link.

CyberPulse framework is implemented as a separate application in SDN application plane. It penalizes the attack flows by dropping the malicious flows and eases the network from flooding attack. The flood mitigation module mitigates the LFA while not interrupting the normal traffic. Finally, it facilitates the traffic to pass normally while constantly checking the link for further attacks. We explain the detail of the submodules in the Incoming subchapters.

### *3.2.1 Link Listener Module*

We collect realistic traffic using an SDN-based mininet network and the floodlight controller. Link Listener is continuously listening to the control channel and collects the network status information. Link Listener works on the basis of an algorithm that runs in the Link Listener. If it finds any congestion it invokes Flood Detection Module. This module continuously collects network traffic statistics and send them to the Flood Detection Module. Link Listener collects network flow statistics and continuously send to the Flood Detection Module that will perform the flood traffic detection process. The Link Listener Module performs the end-to-end measurements on the control channel. The available techniques stress the need for the measurements mechanisms to be employed on both ends of the links. However, Link Listener Module in CyberPulse does not need to install specialized hardware for the traffic measurements.

Link Listener Module inspects the key metrics in the network traffic which are given below.

- Utilized Bandwidth
- Lost Bandwidth
- Packet Drop Rate
- Packet Size
- Full Bandwidth
- Packets Received
- Percentage of packet lost Rate
- Packets Lost
- Percentage of lost byte rate
- Transmitted Byte
- Packets received rate
- Flooding Status

Some statistics are directly computed and some of them depend on other statistics. We define the network statistics in the following.

**Utilized Bandwidth:** It is the percentage of traffic that has been utilized. It is the average rate of successful data transfer through a communication network. The utilized bandwidth is computed by the equation (3-1).

$utilization\ (\%) \frac{=(data\ bits \times 100)}{bandwidth \times interval}$……………………………………..……………….………… (3-1)

**Lost Bandwidth:** The bandwidth lost is the maximum bandwidth subtracted from the utilized bandwidth. The lost bandwidth can be computed by equation (3-2).

$Lost\ Bandwidth = Maimum\ Bandwidth - utilized\ bandwidth$………………………..……… (3-2)

**Packet Loss Rate:** Packet loss occurs when one or more packets are unable to reach the destination. It usually occurs by data transmission errors or congestion on the network links. It is measured by the percentage compared with the packets sent. The TCP identifies the packet loss and performs retransmission to ensure reliable delivery.  It can be computed by equation (3-3).

$Packet\ Loss\ Rate = \frac{Number\ of\ packets\ lost}{Number\ of\ packets\ sent}$ ……………………..………..…………………...……. (3-3)

**Packet Size:** Everything in the internet ecosystem involves packets, for example, every web page on the internet is transferred in the form of packets. Network packets carry information that helps it to reach the destination. The packet contains control and data information. The control information is used to deliver the payload, for example, network address error detection code, and sequencing information.

**Full Bandwidth:** It is the maximum rate of data transfer across a given data path it can be characterized between digital, data, and network bandwidth.

**Packets Received:** The number of packets received at the destination is called packets received.

**Percentage of packet lost rate**: It is the percentage of packets lost in the network during a particular time.

**Packets Lost:** It is the total number of packets lost in a given time. It is occurred due to network congestion or errors in the network.

**Percentage of lost byte rate:** The number of bytes lost in a given period of time.

**Transmitted byte:** The number of bytes transmitted over a period of time.

**Packets received rate:** The number of packets received at the destination.

**Flooding Status:** It is the status of flow on the network. It depends on all the previous factors discussed. It can be flooding or normal. If a flow has a higher packet loss rate, higher bandwidth utilization and lost byte then the status will be flooding. Otherwise, normal status is assigned to the flow.

### 3.2.2 *Pattern Matching Component*

Efficient and expressive pattern matching is a key enabling technology for a complete realization of AmI and humanized computing. This component is encapsulated in the Link Listener Module, it works by maintaining a flood traffic signature database. The traffic is continuously monitored and flows are constantly matched with the signature database. The JSON-based pattern matching engine utilizes REs to match the network traffic with the set of known attack patterns and virus signatures. Known traffic signatures are mitigated here and the unknown signatures are forwarded to the Flood Detection Module. When a flow is found to be malicious, it is forwarded to the Flood Detection Module which classifies the network traffic and adds the malicious flow to the malicious pattern database. Hence, the traffic is efficiently classified.

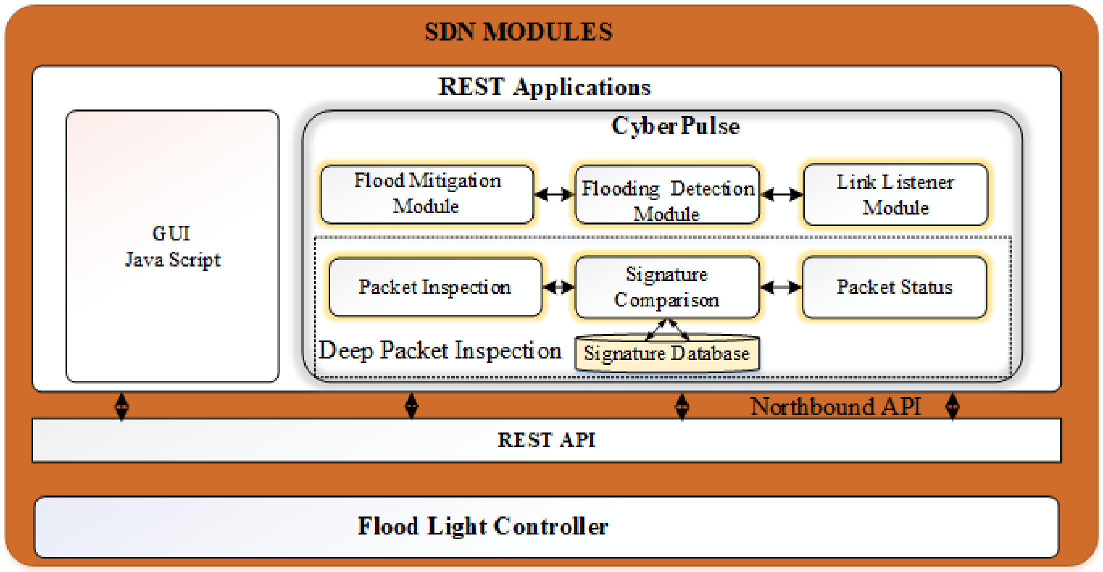


Figure 3-2 A conceptual framework for the proposed solution.

### 3.2.3 *Flood Detection Module*

Flooding occurs when an adversary manipulates bots and send low-rate traffic over the network. The flooding causes higher packet delay, packet loss, bandwidth decrease, and a higher number of bytes transmitted. Therefore, flood detection is vital before performing any mitigation procedure. The flood detection analyzes the occurrence of congestion as well as the location of the congestion. The flooding is detected using the collected metrics using the Link Listener Module. In contrast to the anomaly detection, this module relies on flood detection using the ML classification technique. The Flood Detection Module classify the network traffic using binary classification. The extracted features are based on the SDN specific traffic paradigms. Flood Detection Module is invoked by the Link Listener. If the Link Listener finds any congestion on the link it informs the Flood Detection Module which performs two tasks. First, it inspects the type of congestion on the link, if the congestion is normal and due to benign traffic than the network is allowed to perform its normal flow of operation. If the congestion is caused by an attack, it must be mitigated. At this point, Flood Detection Module invokes the flood mitigation module. The Flood Detection Module performs the traffic classification using the ML technique that is trained using BHP training dataset. The network statistics collected by the Link Listener Module are utilized to identify the malicious and legitimate flows.

### *3.2.4 Flood Mitigation Module*

Flood Mitigation Module receives the classified traffic provided by the Flood Detection Module. It then eliminates the flood traffic using NULL routing or dropping malicious flows and enables the network to carry out the normal flow of operation. It also balances the network traffic after the flood mitigation process. These three modules constantly interact with each other to perform the flood mitigation operation.

Fig 4-1 illustrates a broader architecture of the CyberPulse illustrating components of the proposed architecture. DCI provides the first line of defense against the security threats. It manages a signature database which stores the malicious traffic patterns. The process starts with the Link Listener being the first module to interact with the link. The Flood Detection Module is invoked if there is any congestion observed. Subsequently, the detector module analyzes the type of congestion. If the congestion is caused by normal traffic then the network is allowed to perform the normal course of action and flow of the operation is transferred to the Link Listener again.

The Flood Mitigation Module is called on if the flooding is caused by malicious traffic. This module eliminates the malicious flows and balances the traffic flow over the network. Hence, with the collaboration of all the components, LFA is eliminated and the network is able to perform the smooth operation.

The LFA mitigation framework resides on the application layer as a separate application. With the collaboration of all three modules, the successful operation of the LFA mitigation framework becomes possible. These modules work according to their specification and provide the results to the framework, which ultimately responds to the link congestion or attack successfully.

## 3.3 Design Considerations for the Framework

The following design considerations are important for the implementation of the proposed framework:

- For eliminating LFA on the control channel an application is developed that resides independently of the control or data plane and keeps a check on the link.
- CyberPulse is developed as an independent application and hence, poses a very less overhead in modifying the complex functionality of default SDN controllers.
- To minimize chances of the application crashing, CyberPulse is developed in a way that it doesn't interact with the outside world, so there are fewer chances of its failure.
- Traffic consistency is random, so the framework is scalable according to the incoming traffic.
- The Link Listener Module is able to gather network statistics at random time intervals.

## 3.4 CyberPulse Modules and Working Principle

To address the problem discussed in chapter III, we propose CyberPulse an extension application module to secure SDN [181]. The prime purpose of CyberPulse is to detect and eliminate LFA on the control channel. In this chapter, we, propose an overall detailed architecture of CyberPulse.

We provide broader modules of CyberPulse in the previous section. In this section, we provide a detailed overview of the CyberPulse framework including modules and sub-modules, architectures, and dependencies. To address the problem of LFA, we propose a novel security framework CyberPulse which is developed at the application layer of the SDN plane. The core purpose of CyberPulse is to detect and mitigate LFA on SDN. Figure 3-3 presents the architecture of CyberPulse security solution. CyberPulse contains six modules which are as follows:

1 Configuration Manager

2 Statistics Measurement Manager

3 Stat Measurement Module

4 ML Module

5 Network Status Monitor

6 Flood Defender

The security solution for LFA detection and elimination is based on extracting traffic statistics on a specific interval. These statistics represent samples of the current traffic flow. CyberPulse utilizes supervised ML model which can be trained on 10 different classification algorithms.

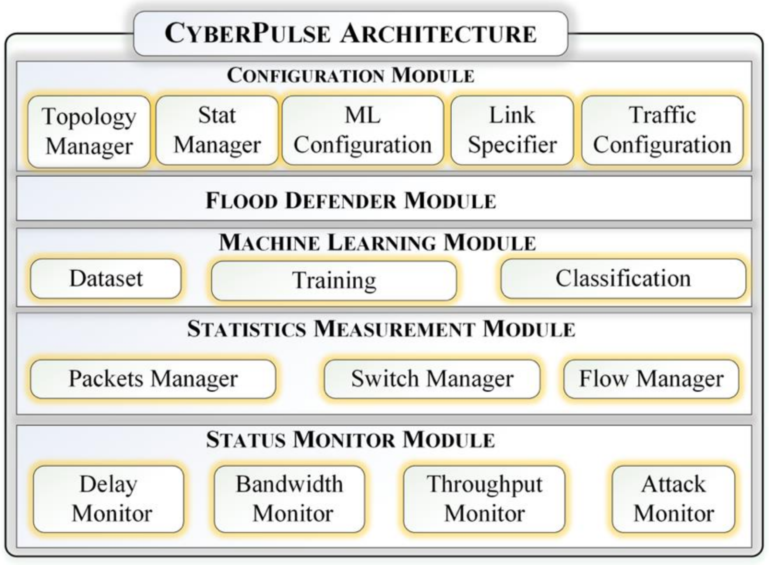


Figure 3-3 CyberPulse architecture.

### *3.4.1 Configuration Manager*

The configuration manager module incorporates multiple sub-functions which are used to input and modify the network configurations. The configurations are provided using a *JSON* configuration file. This configurable file provides flexibility to controls the network topology, statistics management, ML configuration, link specifications, and traffic configuration. All these configurations are alterable in this architecture. The topology manager takes care of the network to use in the experiment. Similarly, the controller configuration sub-module handles the controller parameters, like controller port number and IP address. The statistics manager defines the program duration, statistics collection interval, and flooding threshold of flows in the network. ML configuration module selects the ML algorithm to use, and action to be taken with the flow involved in the flooding of the network traffic. It also specifies the confidence interval for classification. Link specifier denotes the particular link and the switch port number to which this link is attached. The traffic configuration module denotes the link capacity, number of switches, number of hosts per switch, traffic generation server, and IP addresses of the hosts in the network.

### *3.4.2 Statistics Measurement Module*

With the arrival of a flow at the OF switch, the flow entries are matched with the flow table of the switch. The flow is forwarded to the destination as per the flow entry in the switch if the flow entry is not found, it is forwarded to the controller using PACKET_IN message which creates a flow table entry in the switch for forwarding. A Flow_REMOVED message is sent to the controller to remove the flow from the flow table. In addition to this, a thread continuously runs on the controller and queries at a fixed interval the switch's statistics on each flow. In response, it receives a FLOW_STATS message..

CyberPulse aggregates all the statistics for precise classification. Periodic collection of statistics does not impose any overhead as it is basic functionality of SDN architectures and is exposed through southbound interface. Additionally, the classification is performed in parallel by CyberPulse, which have no effect on the performance of the SDN controller by adding any extra functionality.

The statistics measurement manager defines all the statistics to collect during LFA. We collect 14 different statistics as provided in Table 3-2 to detect and mitigate LFA. We utilize a floodlight controller for SDN simulation which provides an interface to collect flow statistics. This module specifies the statistics that are to be collected during the network operation. It includes the schedule of statistical measurement and traffic generation source selection. This module also specifies the information of the Iperf server and client. Stat collector incorporates further low-level modules which are as follows:

- Switch manager
- Flow manager
- Packet manager

These modules are discussed in detail below:

#### 3.4.2.1 Switch manager

This module manages the switches connected to the controller; it first assigns DPID to every switch. It gets the switches list from the '/wm/core/controller/switches/json' json API exposed by the floodlight controller [182]. It then keeps a record of all ports in the network. The switch manager checks the importance of a switch and the port on the basis of how much flooding is being performed by the switch and the port. The switch-id, flow-id is extracted and the host connected to that particular switch is blocked by dumping the flow rule. Switch manager also defines the bandwidth of a particular switch and all the switches connected to the network.

#### 3.4.2.2 Flow Manager

The flow manager reads all the flows in the network and it uses '/wm/core/switch/all/flow/json' json API to manages these flows. It assigns class to the flows on the basis of predefined threshold based on byteCount statistics.

#### 3.4.2.3 Packet Manager

It extracts the packet statistics in the network using the *json* API '/wm/core/switch/all/port/json'. It computes all the transmitted and received packets on all the ports of the switches and calculates the packet loss by subtracting received packets from the transmitted packets. In the same way, it calculates the total bytes, average bandwidth, utilized, bandwidth, packet lost, packet size, packet received, packet received rate, packet loss rate and lost byte rate.

### *3.4.3 ML Module*

Subsequent to statistics collection, this module applies the ML algorithm to identify the flows involved in the LFA. It identifies all the flows in the network and perform the classification on the basis of provided confidence level. ML module can select one algorithm from the available 10 algorithms, the available algorithms are provided in the form of *.pkl* file.

Table 3-2 The extracted features and definitions in the training dataset.

| Feature | Definition | Feature | Definition |
|---|---|---|---|
| Node | Node generating flow | Used Bandwidth | This is what each node could reserve from the reserved bandwidth |
| Utilized Bandwidth | Bandwidth utilized by the node, normalized of bandwidth used | Lost Bandwidth | The amount of lost bandwidth by each node |
| Packet Drop Rate | Packets dropped per unit time | Packet Size | Size of packets on the network |
| Full Bandwidth | Bandwidth of the network | Packets Received | Received packets in a given time interval |
| Percentage of packet lost Rate | Percentage of packets lost | Packets Lost | Packets lost in a given time interval |
| Percentage of lost byte rate | Percentage of lost byte rate | Transmitted Byte | Total bytes transmitted per unit time |
| Packets received rate | Packets received per unit time | Flooding Status | Percentage of flood per node based on Packet Drop Rate |

### 3.4.4 *Feature Extraction*

The advantage of SDN is that OF switches can send statistical information per flow to the controller. The extracted features and their definition in given in Table 3-2. The

- The statistics are measured after every 5 seconds, however, can be adjusted using the configuration file.
- The traffic run time duration is adjustable and it can be set from 1 min or more.
- The program runtime duration is also adjustable.
- ML parameters can be adjusted, and classification model can be used out of 10 available *.pkl* files.
- Any malicious host involved in flooding of the network is dropped.
- The confidence level with which the classification is performed can also be adjusted, any value can be selected from 0-100.
- Network configuration is also adjustable where link capacity, switch count and hosts per switch can be provided.

The ML engine has a pre-trained module which is trained using 10 different algorithms. The pre-trained module is trained in advance to classify the occurrence of LFA.

### 3.4.5 *Network Status Monitor*

The network status monitor updates the real-time network information in the form of graphs. This is the core module of the network because it offers the real-time network to the users. This module provides one of the novel solutions to the ongoing traditional defense mechanisms to monitor network activity at real-time. The network status monitor draws the following graphs.

*1* **Flooding rate per unit time:** This graph demonstrates the link flooding status with respect to time. It presents the link bandwidth status under the attack. Being link and host-specific, it can reveal the intensity of flooding on the desired link and the hosts. It can also be flow specific where it illustrates the level of flooding on the network. Regarding flooding rate, we are talking about the data plane, not the control plane. Also, typically switches themselves are not the terminal points but lead to hosts/servers and we consider switches leading to important devices as the important switches compared with those connecting to end stations. This graph will demonstrate how much data is passing through important links i.e. on links which are connected to switches, to servers or other important devices. Strategic links are the fundamental idea of this research i.e. some links are more important than others.

2 **Packet drop rate per unit time:** This graph describes the packets dropped in unit time.

3 **Network throughput with respect to input load:** There should be both, aggregate throughput for the network and also individual for the links connected to switches connected to servers. For the individual links, you can aggregate throughout them and also illustrate them individually as well.

4 **Traffic delay with respect to input load:** It is the delay incurred on transferring files during LFA.

5 ***Bandwidth saturation with respect to attack rate***. This is to describe the impact of bandwidth saturation with the increase of attack.

6 **Attack detection time with respect to a number of attackers:** This describes the attack detection time of CyberPulse in the presence of attackers.

### *3.4.6 Flood Defender*

The flood defender module eliminates the malicious flows by dropping them. This is implemented as a function in CyberPulse which takes statistics of an active flow (e.g. source IP, destination IP, switch ID, flow ID) and compares the confidence level of the flow with the threshold value set in the configuration file. If the flow has the confidence interval greater than the defined in the threshold, the flow is dropped.

CyberPulse exploits the northbound REST API of the SDN controller to detect and prevent LFA [69]. It is worth mentioning here that CyberPulse detects and mitigate LFA traffic flows that are active in a current session of the SDN. It works concurrently with other modules of SDN in order to perform a seamless operation. CyberPulse incorporates three modules, i.e. Link listener Module, Flood Detection Module, and Flood Mitigation Module. CyberPulse uses the REST API to connect with the control module of the SDN and in-turn the controller uses Southbound API to communicate with the data plane switches [183].

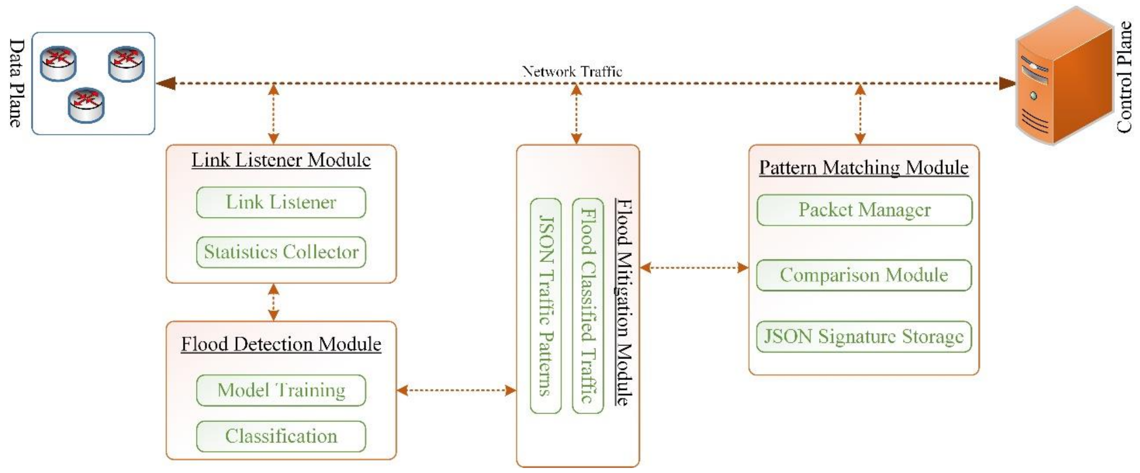


Figure 3-4 Modules of CyberPulse.

It can be observed from Figure 3-4 that there are three modules in CyberPulse, every module performs a specific operation in order to accomplish the cumulative task of LFA mitigation. Fig 4-3 illustrates the flow diagram of CyberPulse process. The process starts with the Link Listener Module inspecting the control channel and continuously sending network statistics to Flood Detection Module. Flood Detection Module inspects the statistics and performs the flow classification. Flood Detection Module incorporates the statistics pre-processing component which eliminates the packet headers information like ACK, SYN-ACK packets from the statistics and presents only the clean traffic flows to the detection module. These statistics are forwarded to the Flood Detection Module which incorporates a ML sub-module, the extracted statistics and the units are presented in Table 3-3. The ML module uses ANN technique to classify network traffic.

Figure 3-5 illustrates the steps involved in the CyberPulse implementation. The figure demonstrates that the network statistics are captured using REST API, subsequently, data pre-processing is performed. Then the pre-processed traffic flows are provided to the ANN classifier which classifies the legitimate and flooding flows. Finally, malicious traffic is mitigated using malicious traffic mitigation module. Similarly, algorithm 1 demonstrates the traffic classification method into benign and flooding flows. This algorithm takes the input of host no, packet statistics, and the training dataset. It extracts the traffic features and classifies the traffic into benign and flooding flows.

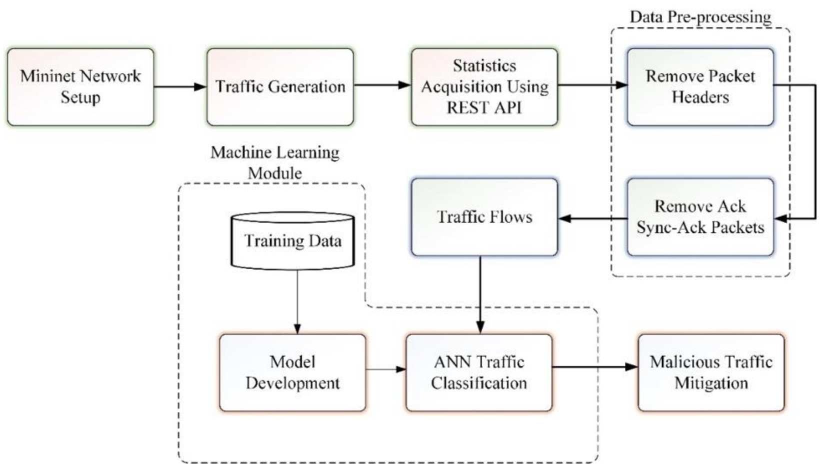


Figure 3-5 Flow diagram of CyberPulse operation.

The ANN classification module builds a training model by using flooding attack dataset [184]. Subsequently, based on this trained model, it performs the classification process. The classifier outputs the benign flows and the attack flows. The results of the classification are forwarded to the Flood Mitigation Module which drop the attack flows using NULL routing technique.

## 3.5 Chapter Summary

SDN has been widely adopted for the datacenter network provisioning and service orchestration. Most of the current LFA mitigation techniques employ link inspection, traffic engineering, and SDN-based approaches. There are very few available techniques that utilize ML to detect LFA. The available ML techniques perform the detection operation in an offline manner. Due to the rapid growth of smart infrastructure, there is a prime need for real-time attack detection in SDN. Therefore, in this chapter, we proposed a security framework CyberPulse which employ ML-based LFA detection and mitigation mechanisms in SDN. Most of the available techniques on LFA detection provides the defense in a reactive manner, however, we have developed CyberPulse to perform LFA defense at real-time.

Although SDN provides efficient virtualization services and facilitates the management problem of the networks. The central controller is the lucrative target for the adversaries. Moreover, the control channel can be attacked by DDoS and LFAs which disconnect the controller from the data plane. We formally presented the problem statement and assessed the feasibility of the ANN technique for flood traffic classification. A holistic approach has been used to mitigate the LFA on the SDN. We introduced two defense mechanisms against the LFAs. The first defense is the pattern matching engineering where we save the patterns of the LFA traffic. We first perform the experiments on the LFA patterns and assign those patterns to the pattern matching engine. The pattern matching engine filters the network traffic from the LFA patterns which can reduce extra overhead on the higher amount of traffic classification.

The final defense is the binary classification based on ML techniques. The ML technique employs Link Listener, Flood Detection, and Flood Mitigation Modules. The purpose of the initial assessment using MLP was to ascertain the feasibility and characteristics of ML in a way that a holistic solution can be developed later. Deep neural networks are one of the most precise classification techniques used in current state-of-the-art problems definition. We have used one ML technique as presented in this chapter and analyzed the core requirements needed to develop a proactive solution against LFA.

# CHAPTER 4. DESIGN AND DEVELOPMENT OF DEEP CONTENT INSPECTION ENGINE

This chapter provides a detailed discussion of the DCI using JSON. DCI provides a first line of defense in detecting LFA. It utilizes JSON-based pattern matching technology to compare the network traffic with the available traffic patterns. It performs this operation by storing a network traffic signature database. This database contains the network flow characteristics such as benign, and malicious flows. The incoming traffic is then matched with the signature database in order to filter the network traffic. So, in this chapter, a detailed background study is performed on a JSON-based pattern matching engine.

In general, pattern matching can be classified as exact (fixed) string matching and approximate matching through REs. In exact string matching, a pattern of a fixed length is matched against a data stream to report all the occurrences of the pattern. Patterns are extensively used to filter network traffic to determine the events of user interest. Data can include wide-ranging formats such as HTTP data and web traffic or network packet captures. Pattern matching commonly occurs in the embedded device sensing the physical properties of the environment such as cameras, motion detectors, temperature, and light detectors or in the Cloud where the data is aggregated. DCI and IDS/IPS also rely on identifying malicious content against defined patterns. In all these cases, the language used needs to be expressive enough to parse a variety of patterns as well as should be efficient in terms of space and time requirement. Therefore, we present DCI in this chapter.

DCI is a packet filtering technique where the data and the header part of the packet are inspected, identifying any abnormality in the packet according to the malicious source, virus, flooding flow, and spams. It is a strong packet filtering mechanism which can also provide the information on whether the packet has been redirected. The packets are passed through the checkpoints in the DCI. These checkpoints compare the network traffic with the rules assigned by the administrator. DCI inspects the content of the packets and analyzes the sources and content of the traffic packets. The traditional packet filtering techniques only analyze the header of the packet which is less sophisticated and trivial. Alternatively, DCI can perform intrusion detection and prevention with key enabling technologies. It is usually used as a firewall for the packet filtering purposes any traffic that passes through this firewall will be clean and conform to the network requirements as the discrepancies will be filtered at the firewall. In this chapter, we will discuss in detail about the DCI techniques and how DFA technology can be used to filter the traffic in DCI. We will discuss different algorithms of DCI and perform the experiments to demonstrate how the DFA-based DCI engine performs exceptionally better in filtering the LFA traffic.

## 4.1 Deep Content Inspection for Traffic Filtering

Before REs, exact strings were used extensively in different applications and some applications even use them now e.g. Snort [57] (an IDS) uses PCRE (Perl Compatible REs) for its pattern matching rules especially in its pre-processing stage to speed up the whole matching process [14]. A number of algorithms exist for fixed string pattern matching, which makes use of character comparison, DFA or NFA etc. [187], [188], [189], [190], [191], [192], [193], [194], [195], [196]. In the past few decades, patterns are widely expressed as REs which are mostly implemented either using DFA or NFA. Each implementation has its own pros and cons. However, in this research, we have focused only on DFA-based pattern matching. In the worst case, DFA has processing complexity of O (1) for each input character whereas its space complexity is *O (n)* (refers to alphabet on which DFA is defined and *n* refers to the length of the RE).

In this domain, most of the previous works are either focusing on increasing pattern matching, the speed or on reducing memory consumption. Recently, some attempts are utilizing Graphics Processing Units (GPU) and multi-byte pattern matching techniques to achieve better matching speed [59], [62], [63]. In [197] the authors discuss high-speed automata processing, and their implementations on a variety of parallel platforms: CPUs, GPUs, Field Programmable Gate Arrays (FPGAs), Application-specific integrated circuit (ASICs), and network processors. Moreover, many algorithms have also been proposed to reduce the memory footprint using alphabet reduction, RE rewriting and grouping techniques [61], [64], [66], [67], [58], [198], [57], [199], [200], [201], [202], [203], [204], [205], [206], [207], [208], [207]. Tsai et al. [209] propose a simultaneous pattern matching methodology for wildcard patterns by two separated engines to represent discrete finite automata for improving the performance of IoT network security. Wang et al. have pointed out that the problem of state explosion does not lie in DFAs. In fact, the root cause of the problem actually lies in the REs as it can be seen that no such problem exists in the DFA of exact strings [210]. A DCI set up in the CyberPulse has been demonstrated in Figure 4-

1. The figure illustrates the CyberPulse set up in providing surveillance on the control channel.

The DCI is acting as the first line of defense against LFA where the traffic first arrives at the packet inspection module. The signature comparison module compares the incoming packet with the packet database. If it finds a malicious packet then it drops the packet and adds the signature in the database. Otherwise, it forwards the packet to the Link Listener Module which performs the normal flow of operation using the traffic classification and mitigation. The following are some of the factors that are primarily responsible for increasing parsing time of REs that eventually leads to time and space complex DFAs.

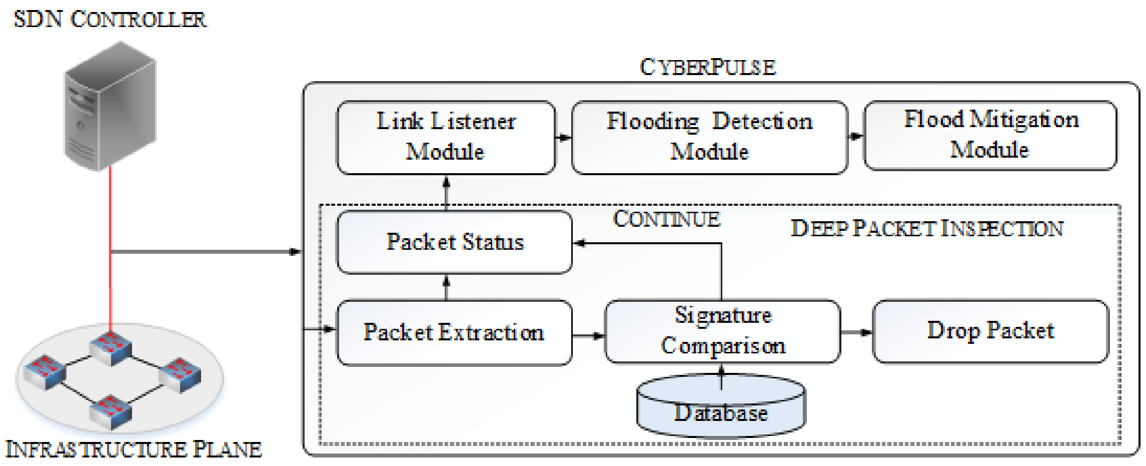


Figure 4-1 Deep content inspection-based traffic filtering against LFA.

### *4.1.1 Semantic Overlapping for Attack Traffic Pattern Matching*

In real-world REs, character sets are often observed with Kleene closure (e.g. “[a-m]*") or counting constraints (e.g. “[a-z] {m, n}”). A counting constraint gives the number of times a certain subpattern in a given RE can be repeated e.g. sub-pattern {min, max}. Here, “min" and “max" specifies the minimum and a maximum number of times a sub-pattern can be repeated respectively. In [211], Kleene closure and counting constraint are termed as overlapping factors. Most of the times, overlapping factors are considered to be one of the major reasons for constructing time and space complex DFAs. For example, we have two patterns, RE1, and RE2, where RE1 is “.*ab.*cd.*mn" and RE2, is “.*efgh". Here, RE1 comprises of two overlapping factors (“.*") such that the composite DFA of RE1 and RE2 requires RE1 to replicate states of RE2 sub-DFA at each of its overlapping factors as RE1 can match any input string that matches with RE2 as represented in Fig 4-2. Composite DFA [212], [213] is a basic terminology in Automata theory. For a given RE, the first (last) fragment is called its eldestson (youngestson), correspondingly other fragments are non-eldestsons (non-youngestson).

To match multiple REs together in a single pass, all the eldestsons are compiled into a composite DFA, and each non-eldestson is compiled into an individual DFA. Thus, there exists a semantic overlapping between the overlapping factors and the patterns, which eventually leads to state explosion. Since REs used in real pattern matching engines comprise of a large number of such overlapping factors, therefore, the exponential increase in the number of DFA states is often seen. Yu et al. proposed RE rewrite rules for a certain type of REs, which can limit the size of the DFA as well as the parsing time of REs [57]. Thus, the complexities present in real-world REs especially the presence of repeated counting constraints and dot star terms cannot be dealt with pure DFA-based solutions [214]. In [58], a hybrid finite automata solution is proposed to cut down the state blowup problem. The hybrid approach plays a critical role during subset construction where it disrupts the NFA to DFA conversion process at those states of NFA, which otherwise lead to state explosion.

Table 4-1 Impact of semantic overlapping on DFA states based on RE type.

| **Types** | **RE Types** | **Number of states** |
|---|---|---|
| Type1 | $\mathrm{ES1}.\varPhi\ *.\mathrm{ES2}$ | No/constant state inflation |
| Type2 | $\mathrm{ES1}.\varPhi\ *.\mathrm{ES2}$ | Exponential state explosion |
| Type3 | $\mathrm{ES1}.\varPhi\{m,n\}.\mathrm{ES2}$ | State explosion |
| Type4 | $\mathrm{ES1}.*\varPhi\{n,\}.\mathrm{ES2}$ | No state inflation/linear or polynomial state inflation |
| Type5 | $\hat{}\mathrm{ES1}.*.\varPhi\{n\}.\mathrm{ES2}$ | Polynomial state inflation |
| Type6 | $U_{k=1}^{n}(ES_{k1}.\varPhi^{*}.ES_{k2})$ | Exponential state explosion |

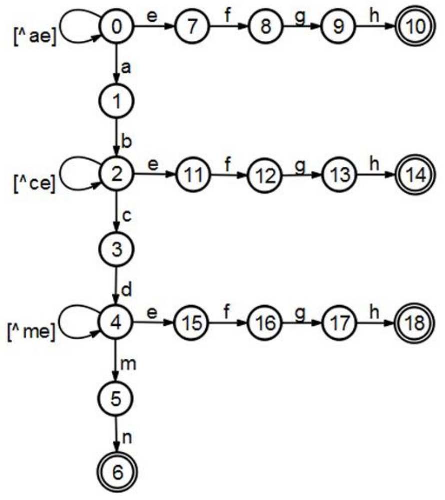


Figure 4-2 DFA representing.*ab.*cd.*mn and.*efgh (Some transitions are not presented due to clarity).

In [26], a study was conducted on six different types of REs of the form ES1. OF.ES2 to view the impact of semantic overlapping on states of the corresponding DFA. The results are briefly depicted in Table 4-1 where "ES" refers to exact substring, "$\Phi$" refers to union set of characters from the alphabet, "m" and "n" are lower and upper limits of the counting constraint respectively. It can be observed that Type 2, 3 and 6 can lead to DFA state explosion due to semantic overlapping. The interested reader can refer to [210] for the detailed quantitative description of each type. Figure 4-2 illustrates an example of DFA *ab.*cd.*mn and.*efgh.

### ***4.1.2 Traffic pattern Grouping Techniques***

As discussed earlier, composite DFA is greatly preferred as compared to individual DFAs due to its lower processing cost however there are certain situations in which creating a composite DFA of a set of REs become infeasible. There are two well-known reasons behind this infeasibility:

- The composite DFA gives rise to state explosion and storing such a large DFA in memory has a very high cost.
- The parsing of REs takes too long for composite DFA construction.

The notation of RE is a bit ambiguous as the same pattern can be represented in different regular REs with varying efficiencies. The presence of even a single inefficient RE in a group of REs often puts a negative impact on the compilation phase for the construction of the DFA. For example, the compilation phase can either take an optimal time or it might take extremely long (exhausting system resources) depending upon the REs present in a group. Therefore, some authors have proposed grouping algorithms to figure out inefficient REs and problematic interactions between the REs before making groups [57], [210]. In fact, overlapping factors of different patterns can interact with each other in a certain way that prolongs the time to implement their DFA (as in the case of exponential state inflation). In [210], these problems are addressed by splitting all the REs in the given RE set into two categories. Each category comprises of segments of REs such that the segments in the same category do not have semantic overlapping. Thus, small changes in REs can bring an overall positive impact on the compilation (parsing) phase.

### *4.1.3 Alternative Approaches to Pattern Representation*

In the recent past, some alternative approaches to pattern representation have been proposed such as Open Data Description Language (OpenDDL) [215], Human-Optimized Config Object Notation (HOCON) [216], YAML [217], and MongoDB [218] that are based on JSON. While this research only focuses on pattern matching based on RE a comparison is presented here with these competing approaches for the sake of completeness.

The OpenDDL is a text-based language for storing or exchanging generic data in a concise human-readable format. The data in an OpenDDL file is explicitly typed to eliminate ambiguity and fragile inference practices that can impact the integrity of the data. This strong typing is further supported by the specification of the number of bits required to store numerical data values when using binary notation.

HOCON is Human-Optimized Config Object Notation and is a human-readable format for storing data. It is a superset of JSON and is supported by several frameworks such as Play Akka.NET and Puppet. YAML is a human-readable data serialization language commonly used for configuration files. It supports various applications that require storage or transmission of data, for instance, debugging output and document headers. YAML 1.2 is a superset of JSON. Document databases such as MongoDB use JSON to store records, just as tables and rows store records in a relational database. Internally MongoDB represents JSON documents in a binary-encoded format called BSON. It is an extension of the JSON model for the purpose of having additional data types, ordered fields, and to gain efficiency in encoding and decoding within different languages. A JSON database returns query results that can be easily parsed, with little or no transformation, directly by web programming languages such as JavaScript.

## 4.2 Pattern Matching Engine for Traffic Filtering

Patterns are extensively used to filter data produced by the networking devices to determine the malicious traffic in a network. Data can include wide-ranging formats such as HTTP data and web traffic or network packet captures. Pattern matching commonly occurs in the embedded device sensing the physical properties of the environment such as cameras, motion detectors, and temperature and light detectors or in the cloud where the data is aggregated. DCI and IDS/IPS also rely on identifying malicious content against defined patterns. In all these cases, the language used needs to be expressive enough to parse a variety of patterns as well as efficient in terms of space and time requirement.

Hence, there exist a number of ways to use patterns in different networking applications. In the past few decades, REs has emerged as the most appropriate way to define textual patterns as they are highly flexible as compared to fixed strings. At a first glance a RE seems to be an encoded piece of message e.g. "^a {2, 4} b*{bc | ae} + [^d]". This lacks human readability is due to excessive use of meta-characters, which have special meaning when they are combined with the literal characters in the RE. At the same time, the flexibility present in REs is also due to the use of meta-characters and counting constraints. Table 4-2 gives a description of some of the well-known meta-characters that are mostly used in REs.

### *4.2.1 JSON Format for Network Pattern Representation*

In this section, we first discuss some of the important notations and keywords that are used in our proposed JSON-based pattern language. The notation is compared with its RE counterpart in order to make a good comparison of the two pattern expression approaches as provided in Table 4-2. All the RE-based notations from 1-11 are expressed using key-value pairs, which is the key characteristic of data representation in JSON. Most of these notations are used to put a restriction on the repetition of characters or digits. For example, A {5, 10} in RE indicates that 'A' can occur a minimum of 5 times and a maximum of 10 times. To express this same case in JSON, a keyword, “Rep_Bound" is introduced. Its value is an array consisting of 2 integer values. Both values indicate minimum and maximum bound respectively for the repetition of character or a group of characters.

In REs, chars/digits represented inside square bracket (character classes) can express two different things depending upon the presence of the caret symbol **“^”**. If this symbol is present inside brackets then any character is allowed except those present inside brackets. Otherwise, if the caret symbol is not present then any single character from the given range of characters can appear. In JSON, the character class is represented by the key, “Range” and its value is an array of three values, where first two values of the array are Boolean and the third one specifies the range of characters or digits. The first value of the array indicates whether the set of character needs to be negated. In case of negation, it is set to true else it is set to false (false indicates any character from the character class can appear). The second value of the array specifies if the set of characters have specific repetition requirement like “zeroormore” and “oneormore” etc.

Table 4-2 RE symbols and their equivalent JSON representation.

| **Sr.** | **R.E** | **JSON** | |
|---|---|---|---|
| | | **Notation** | **Comments** |
| 1 | {m,n} | “Rep_Bound”:[m,n] | Min=m (int) , Max=n (int) |
| 2 | {n} | “Rep_Bound”:[n,n] | Min=Max or m=n |
| 3 | * | “Properties":”zeroormore” | - |
| 4 | + | “Properties”:”oneormore” | - |
| 5 | ? | “Properties”:”zeroormore” | - |
| 6 | . | “ANYCHAR”: true<br>“sANYCHAR”: true | “sANYCHAR" is used in that case where it has associated properties like zeroormore and oneormore |
| 7 | \| | “OR”: True | - |
| 8 | {abc}<br>{abc}+ | “Range”:[“false”,”false”,”abc”]<br>“Range”:[“false”,”true”,”abc”] | “Range”:[Negation (bool), Properties (+,*,?) (bool), range of chars] |
| 9 | {^abc}<br>{^abc}+ | “Range”:[“true”,”false”,”abc”]<br>“Range”:[“true”,”true”,”abc”] | - |
| 10 | ^ | “Start”: true | Pattern should appear at the start of a string |
| 11 | TexttPartofPattern | “Chars” : “TextPartofPattern”<br>“ssChars”: “TextPartofPattern” | \sChars" is used in that case when the chars have associated properties like zeroormore and oneormore |

Furthermore, we have also designed and implemented the architecture that makes use of JSON-based patterns. These patterns are used for inspecting network traffic and its details are given in Chapter 6. Next, the representation of patterns in the JSON format is discussed here using examples.

### *4.2.2 Description of Patterns*

Following is the description of a pattern, which is to be represented in JSON format: (The equivalent RE of this pattern can be expressed as “^a{2,4}b*(bc|ea)+[ ^d]").

- The pattern should be present at the start of a string
- It should start with 'a' that occurs a minimum of 2 times and a maximum of 4 times
- Followed with zero or more occurrences of 'b'
- Then, one or more occurrences of “bc” or “ea”
- At last, the pattern should end with any character other than 'd'

**Example 1:** A pattern comprises of different small sub-patterns or logical blocks. The first step of the proposed format is the identification of these logical blocks. Once they are identified, it becomes easier to write a pattern in JSON without any collision of the same keys as multiple keys with identical names are not allowed within the same structure in JSON. Here, the pattern described above forms four logical blocks. For each block, a value is defined that can be an integer, string, NULL, boolean, array or an object. In our example, a pattern is required to be present at the start of the input string that is specified by the key “Start”, which is set to “true” in the pattern as described below. Then, “B1” is defined which is the next logical block.

Here, the keyword, “sChars” is used for “a” as it has some specific repetition requirements that are defined using “Rep_Bound”. It states that character “a” can occur a minimum of 2 times and a maximum of 4 times. For the logical block “B2”, another object is defined, which states that “b” can occur zero or more times as mentioned by “Properties”. Next, an object is defined for “B3” that also has a sub-object “B 3a”. In the absence of “B 3a", the key “Chars” might have collision inside the “B3" object resulting in invalid JSON pattern. Moreover, keys “GroupStart” and “sGroupEnd” are also used in order to mark a boundary on which the OR operation is valid. Here “sGroupEnd” is used, which demonstrates that the bounded group of characters has associated repetition properties (“one or more” in this example) otherwise “GroupEnd” is used. The last logical block of the pattern is “B4”, where “Negation” is set to true in the array of “Range”. This means that any character other than “d” can appear as the last character of the pattern. Figure 4-3 provides the illustration of logical blocks of the pattern. The Figure 4-4 illustrates the example of pattern representation in JSON.

The first step is again the identification of logical blocks of the pattern. The above pattern comprises of 5 logical blocks (“B1”, “B2”, “B3”, “B4”, “B5”). An illustration of the logical blocks for the same pattern expressed in a RE is depicted in Fig 4-3. The JSON representation of this pattern is also provided. Here, it can be observed that a sub-object “B-5a” is defined for logical block “B5”. This is performed in order to avoid the collision of key “Chars”. As mentioned earlier, the main role of these blocks is to avoid any identical key name collision.

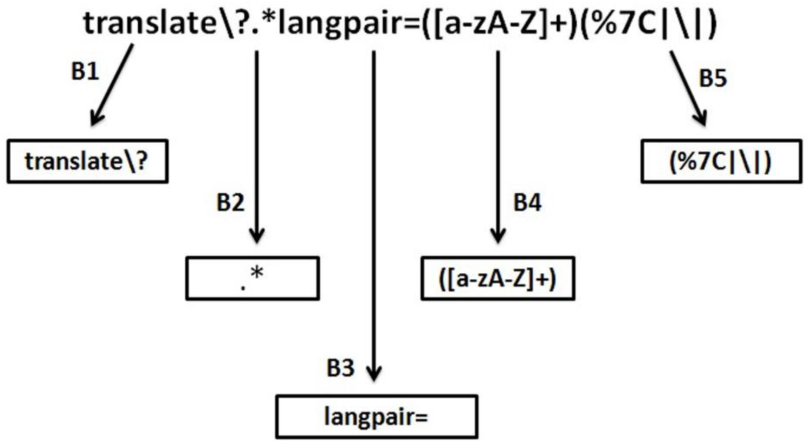


Figure 4-3 Illustration of logical blocks of the pattern.

**Example-2:** The following is another description of a pattern for representation in JSON:

- The Pattern starts with “translate?”.
- Followed with zero or more occurrences of any character.
- Then “langpair=” appears.
- Followed by one or more occurrences of any single character from the range of 278.
- a-z or A-Z.
- At last, the pattern should end with either “|” or “%7C”.

**Example-3:** Here is another example of a pattern, equivalent RE which can be expressed as “.gif\x3f [a-f0-9] {4, 7}\x3d\d {6, 8}"

For JSON representation, the description of this pattern is as follows:

- The pattern can start with any character followed by “gif”.
- Then, the character “?” with position 0x3f in the character set.

- Afterwards, any single character from the range of a-f or 0-9 with minimum occurrence 4 times and maximum 7 times.
- Followed by the character "=" with position 0x3d in the character set.

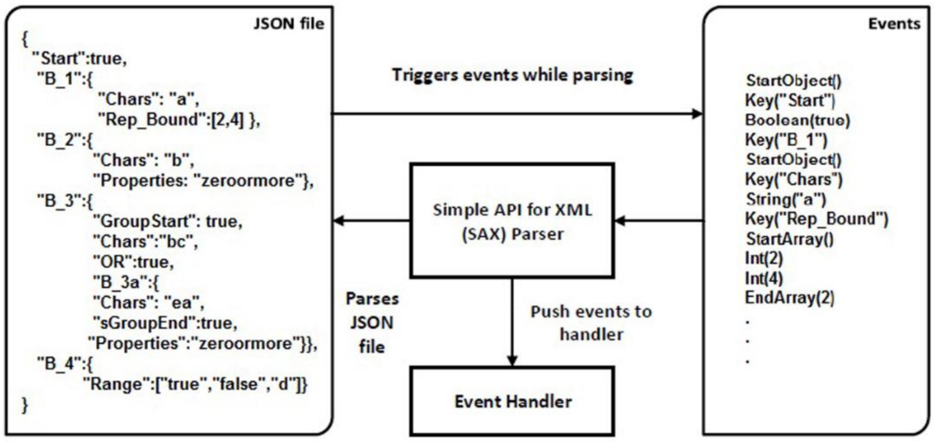
**Example-1:**

```
"Start": true,
"B1": {
"sChars":"a",
"Rep_Bound": [2, 4]
},
"B2": {
"sChars":"b",
"Properties": "zeroormore"
},
"B3": {|
"GroupStart": true,
"Chars": "bc",
"OR": true,
"B 3a": {
"Chars": "ea",
"sGroupEnd": true,
"Properties": "oneormore"
}
},
"B4": {"Range": ["true", "false","d"]}
}
```

**Example-2**

```
{
"B1" : {"Chars" : "translate?" },
"B2" : {
"sANYCHAR" : true,
"Properties": "zeroormore"},
"B3" :      {"Chars" : "langpair="},
"B4" :      {
"Range" : ["false", "true", "a-zA-Z"],
"Properties" :    "oneormore"},
"B5" :      {
"GroupStart" : true,
"Chars" : "%7C",
"OR": true,
"B5_a" :{"Chars" : "|" },
"GroupEnd": true
}
}
```

Figure 4-4 Examples of pattern representation in JSON.



```
{
  "Start":true,
  "B_1":{
          "Chars": "a",
          "Rep_Bound":[2,4] },
  "B_2":{
          "Chars": "b",
          "Properties: "zeroormore"},
  "B_3":{
          "GroupStart": true,
          "Chars":"bc",
          "OR":true,
          "B_3a":{
          "Chars": "ea",
          "sGroupEnd":true,
         "Properties":"zeroormore"}},
  "B_4":{
       "Range":["true","false","d"]}
}
```

```
StartObject()
Key("Start")
Boolean(true)
Key("B_1")
StartObject()
Key("Chars")
String("a")
Key("Rep_Bound")
StartArray()
Int(2)
Int(4)
EndArray(2)
.
.
.
```

Figure 4-5 Parsing of JSON files using SAX-style parser.

Figure 4-5 illustrates the parsing of JSON files using Simple API for XML (SAX)-style parser. At last, the pattern should end with a digit from the range 0-9 with minimum occurrence 6 times and maximum 8 times. Here, three main objects or logical blocks "B1", "B2" and "B3" are defined for the pattern. It can be seen in this representation that there is no special need for object creation for "B1" and "B3". It is due to the reason that there is no keyword in common between the two. As a result, the content of both these logical blocks can be written without the creation of any objects. However, the presence of three logical blocks avoids any ambiguity in the case of large and complex patterns.

Furthermore, a pattern can have different representations in JSON. For example, there is a pattern that starts with either “Reader” or “Leader” followed by any two digits from the range of “0-9”. This pattern can be expressed in different ways but here we discuss only two JSON variants of it. Figure 4-6 illustrates another representation of RE in JSON.

```
Example-3:

“B1”: {
“ANYCHAR”: true,
“Chars”: “gif\\x3f”
},
“B2”: {
“Range”: [“false”, “true”, “a-f0-9”],
“Rep_Bound”: [4, 7],
“Chars”: “\\x3d “
},
“B3”: {
“SChars”: “\\d”,
“Rep_Bound”: [6, 8]
```

Figure 4-6 Another example of RE representation in JSON.

Both representations are efficient as they can represent the pattern in an equal number of states (i.e. 9 states) in DFA. However, if the DFA minimization algorithm is not applied then it is observed that Representation-1 is much more efficient as it can represent a DFA in nearly half of the states (9 states) as that required by the Representation-2 (15 states). This experiment is conducted on our JSON-based pattern matching engine, which is discussed in Chapter 6. Due to the event-based parsing style adopted for patterns and as observed through experiments, it can be observed that JSON patterns take optimal time in the compilation phase and their resultant DFA structures are realizable in memory too. Fig 4-8 illustrates the structure of JSON Pattern Matching Module. The pattern can be expressed in different ways but here we discuss only two JSON variants of it Figure 4-7.

```
Representation-1:
  “Range”: [“false”, “false”, “LR”],
  “Chars”: “Leader”,
  “sChars”: “\\d”,
  “Rep_Bound”: [2, 2]
  }
```

```
Representation -2:
  “GroupStart”: true,
  “Chars”: “Leader”,
  “B1”: {
  “OR”: true,
  “Chars”: “Reader”
  },
  “GroupEnd": true,
  “B2”: {“Range”: [“false”, “false”, “0-9"] },
  “B3” : {“Range” : [“false”, “false”, “0-9”] }
  }
```

Figure 4-7 Representation of RE

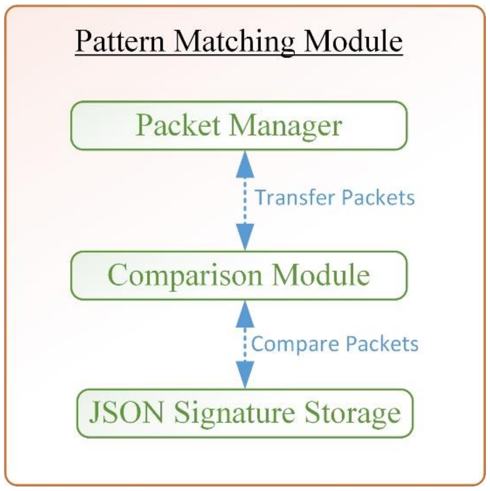


Figure 4-8 JSON pattern matching module.

## 4.3 Comparative Analysis of Parsing Techniques

Parsing is an initial stage of the multi-stage pattern matching process. Once a pattern is defined, it is first divided into tokens, which are then parsed to determine the relationship among the tokens. In case of REs, a binary tree is often built during parsing of the expression. Next, the tree is converted into an NFA and finally into a DFA. Parsing is a significant part of any pattern matching process. Therefore, this chapter reviews different parsing techniques available especially for JSON as we have proposed a JSON approach for pattern representation. Following are three broad classes of parsers.

- DOM-style Parser
- SAX-style Parser
- StAX-Style Parser

### *4.3.1 DOM-style Parser*

The DOM (Document Object Model) allows the parser to load the entire content, which needs to be parsed into the memory. The content is stored there in a tree-like structure before any program starts processing it. Any node of the tree can easily be accessed, modified or deleted as data is always available in the memory. However, this type of parser is not considered a good option when memory resources are low and data is large. In such a case, it can consume a large memory to store the entire content in a tree-like structure. Thus, DOM is only useful when data is small and it often needs modifications.

In our case, the data comprises of JSON-based patterns. This data can be large if a composite DFA of multiple patterns is built otherwise it is small if an individual DFA is built against each pattern. Hence, the memory cost actually depends on the design of the application. Moreover, the capability of the DOM-based parser to access any part of the document easily for modifications/deletions does not add much worth in the present scenario. This is due to the reason that JSON-based patterns do not require random access while parsing for any sort of modification.

### *4.3.2 SAX-style Parser*

SAX is an event-based parser, which reads through the document and triggers an event, for instance, it encounters something like the start or the end of the object or array. The events are pushed to the event handlers in order to take appropriate action. The SAX () parser is based on a push model. An illustration of the scenario using a SAX parser for parsing JSON patterns is also presented in Fig 5-4. Unlike the DOM, the content can be read in small parts without exhausting the memory as there is no need to load the entire document or content into memory. For JSON-based patterns, a SAX-style parser seems to be the most appropriate option due to following reasons.

- The pattern matching engine needs to compile (parse) the set of patterns without random access to any pattern and modification on a certain part of the pattern later once it is processed.
- It can handle both large and small pattern files efficiently even if memory resources are low.

### *4.3.3 StAX-style Parser*

A StAX (Streaming API for XML) parser is based on a pull model in which the program decides the part of the streaming data that needs to be accessed. Unlike SAX, the data is not pushed into the program but is instead extracted by the StAX parser. This is performed based on instructions from the program as parsing is controlled by the application. Furthermore, data is read in small chunks just like in the case of a SAX parser instead of exhausting resources by storing the entire document into memory. Due to the following reasons, a StAX-style parsing is not suitable for parsing JSON patterns:

- Each part of the pattern file is highly significant and no part of it can be skipped (as a StAX parser allows to skip data) while parsing the file.
- The application needs to know the exact structure of the pattern well before parsing as it has to pull the data. However, in our case, every JSON-based pattern will have its own structure and it is not convenient for the application to know the structure of every pattern before parsing.

Figure 4-8 illustrates the architecture of the JSON pattern matching module. There are a number of parsers available for parsing JSON content. A brief comparison of characteristics of some of the well-known parsers is presented in Table 4-3. These characteristics include.

1. **Language:** Language supported by parser
2. **Technique:** The paradigm used by the parser to pass its results to the application. For example, SAX, DOM and StAX etc.
3. **Validation:** Whether or not the parser checks the JSON for validity and syntax errors.
4. **Low memory usage**: Whether or not the parser stores the input or a complete representation of the input in memory. A system whose memory usage is dependent on the length of the input is considered to have high memory usage.
5. **Streaming support:** Whether or not the parser can process JSON from a stream in real time. This implies that the input can be processed without having the entire JSON document on hand.
6. Here, it needs to be mentioned that no particular parser can be labeled as the most efficient as some parsers perform extremely well on small data whereas others perform efficiently on larger datasets. For instance, the DOM-based parser performs extremely well on small data but in the case of large data, a SAX/StAX parser is generally preferred. This is due to the reason that the DOM requires the entire content to be stored in memory that exhausts the resources and is highly inefficient. Thus, each style of parser has its own pros and cons. Therefore, there are many parsers that support a combination of DOM, SAX or StAX in order to provide the optimal performance depending upon the application requirements. However, it appears from all the above discussion that a SAX-style parser is still the most appropriate choice for parsing JSON-based patterns.

Table 4-3 Comparison of relevant parsers.

| Parser | Language | Technique | Validation | Low Memory Usage | Streaming support |
|---|---|---|---|---|---|
| RapidJSON [34] | C++ | SAX/DOM like | Yes | Yes (in case of SAX) | Yes |
| JSONCpp [35] | C++ | DOM like | Yes | No | No |
| JSONlite [36] | C | SAX like | Yes | Yes | Yes |
| YAJL [37] | ANSI C | SAX like | Yes | Yes | Yes |
| Jettison | Java | Stax like | No | Yes | Yes |
| GSON [38] | Java | DOM/Streaming/mixed (DOM+ streaming) | Yes (in case of DOM) | Yes | Yes |
| Jackson [39] | Java | StAX-like, Data binding/DOM like | Yes (in case of DOM) | Yes | Yes |

## 4.4 Experiment for Deep Content Inspection

This section has three main parts. The first part highlights the significance of the compilation phase by comparing it with the matching phase on the basis of different metrics. As mentioned earlier, patterns are generally compiled offline into a DFA and later the stored DFAs are used for pattern matching. However, patterns are nowadays updated very frequently. Therefore, if patterns are compiled in real-time then not only can memory be conserved but also the latest or updated patterns can be used in real-time. Here, we may expect the compilation phase to create a bottleneck. The objective of the experiments in the first part of this chapter is to prove the comparative complex of compilation in time and memory and make a case that research needs to be conducted in this area. The second part of this chapter evaluates the performance of the proposed JSON-based expressions and compares the effectiveness of this new approach with REs. All the experiments have been performed on a system with 2.5 GHz Intel Core I5-3210 M processor and 4 GB of memory. The third part of the evaluation determines the effectiveness of the proposed pattern language through a user study whereby real users comment on the acceptability of the syntax in comparison to competing approaches.

### *4.4.1 Comparative Evaluation of Compilation and Matching Phase*

We performed some experiments to compare the complexity of the compilation and matching phase of DFA-based RE processing engines, which includes RE2, Rexgrep and Regex-Processor [40], [41], [42]. These engines are profiled and instrumented using Perf and Pintool respectively to determine different performance metrics and instruction types as will be discussed later in this chapter [43], [44]. Moreover, a network trace file of size 1.2 GB (obtained from DARPA IDS datasets [45]) is matched against different sets of REs using all the available matching engines. The details of these RE datasets is given in Table 4-4. The datasets ranging from 1-5 comprise of Perl Compatible Regular Expressions (PCRE) from Snort [219].

Table 4-4 RE Dataset usage details.

| **Datasets** | **RE2** | **RE Processor** | **Rexgrep** |
|---|---|---|---|
| snort24 | ✓ | ✓ | X |
| Snort30 | ✓ | ✓ | X |
| Snort40 | ✓ | ✓ | X |
| Snort50 | ✓ | ✓ | X |
| Snort60 | ✓ | ✓ | X |
| random21 | ✓ | X | ✓ |
| Random30 | ✓ | X | ✓ |
| Random40 | ✓ | X | ✓ |

Snort24 through snort60 consist of increasing number of RE patterns and hence the DFA states that are generated also increase. Snort RE datasets present a good candidate for evaluation of our work because pattern matching engines like IDS need to be frequently updated with new patterns. However, patterns expressed using REs are long and extremely complex; snort60 comprises 60 REs for instance. It is tedious to design such patterns and keep them updated. In this regard, if a previously defined RE needs to be updated, then one of the problems is to locate the exact place where changes are required. Secondly, it is not easy to be certain that one has not disturbed the remaining RE while modifying the previously defined RE.

Table 4-5 Description of characteristics of RE datasets.

| **Sr.** | **Dataset** | **No. of RE (patterns)** | **% anchored RE (^,$)** | **% RE using wildcards (*,+,?)** | **% RE with length restrictions {n},{n, m}** |
|---|---|---|---|---|---|
| 1 | snort24 | 24 | 50 | 79.16 | 0 |
| 2 | Snort30 | 30 | 86.6 | 100 | 6.66 |
| 3 | Snort40 | 40 | 70 | 30 | 15 |
| 4 | Snort50 | 50 | 64 | 66 | 26 |
| 5 | Snort60 | 60 | 58.3 | 56.66 | 20 |
| 6 | random21 | 21 | 0 | 42.86 | 4.76 |
| 7 | Random30 | 30 | 0 | 30 | 3.33 |
| 8 | Random40 | 40 | 0 | 22.5 | 2.5 |

In view of these problems, it can be observed that our proposed JSON representation is appropriate for defining large and complex patterns and is also capable of handling quick modifications. This approach allows the creation of logical blocks of the pattern. If a network security admin were to remove certain blocks from the snort dataset defined in the JSON pattern, then it is easier to locate those blocks. Moreover, problems like invalidity of the pattern will not arise after certain blocks in the pattern get removed or modified.

Then, there are three more datasets: random21, random30 and random40, which are created using different real-world RE sources of varying complexities. These random datasets are non-anchored and are fully compatible with Rexgrep as it does not support anchored REs by default and some other important features of PCRE. *Non-anchored* RE is a pattern that has no anchors, ^ for start of string, and $ for the end of string, and thus allows partial matches. In the case of Regex-Processor, each random RE datasets gives rise to more than 100,000 states during the NFA to DFA conversion. This ultimately leads to building up of multiple smaller DFAs for a single RE dataset. However, we are interested in performing all the experiments using a single composite DFA for each given RE dataset. As discussed earlier, multiple smaller DFAs and single composite DFA for a set of REs have different impacts on the parsing and matching speed. Therefore, random21, random30 and random40 are not used for Regex-Processor. A brief overview regarding the usage of all the RE datasets against each of the matching engines is presented in Table 4-4.

First, we measured the time taken individually by the compilation and matching phases for the given RE datasets. In the case of Regex-Processor and Rexgrep, the time taken by the parsing (compilation) phase for each dataset is considerably high as compared to the matching phase. However, the results are extremely opposite in the case of RE2 where the compilation phase takes less time as compared to the matching phase.

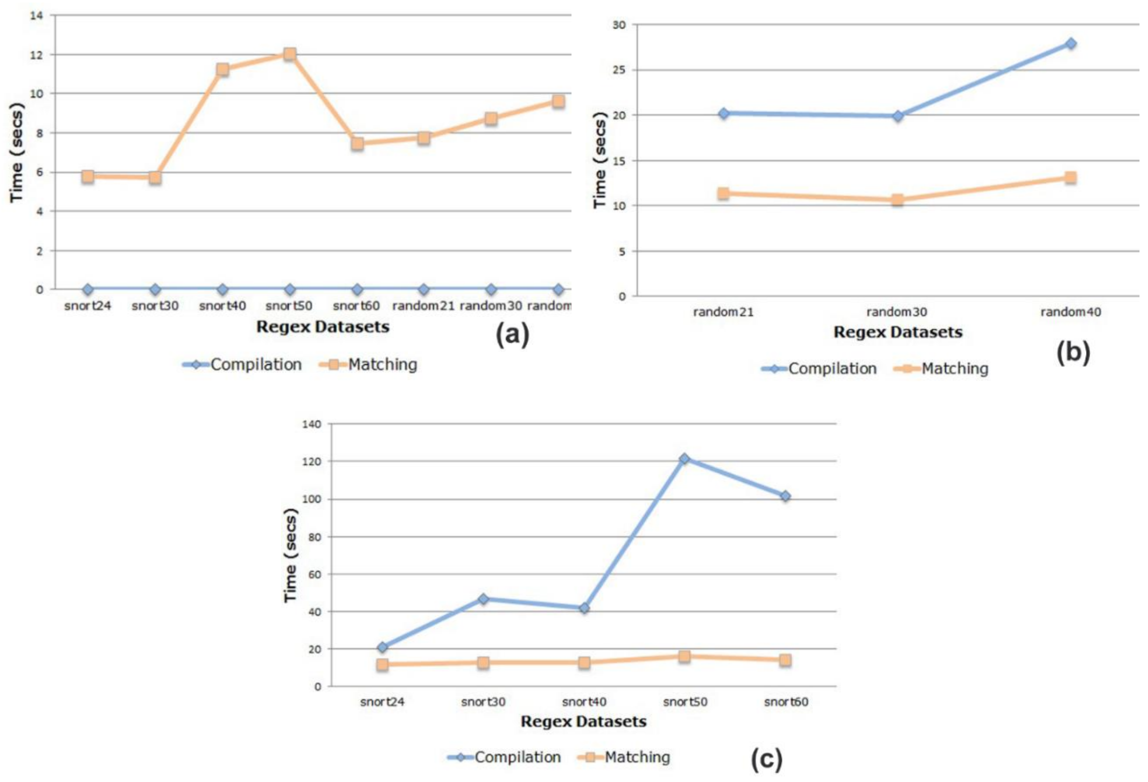


Figure 4-9 Time elapsed in compilation and matching phase. (a) RE2 (b) Rexgrep (c) Regex-Processor.

Figure 4-9 demonstrates the time plots for all the three matching engines RE2 takes 3 milliseconds on average for all the RE datasets, which is the fastest among the three matching engines. Such results from RE2 are due to its unique style of compiling REs. In RE2, a set of REs is first parsed to create a parse tree, which is then compiled into a set or a graph of instructions known as \Prog". Afterward, its DFA engine executes \Prog" to search for any pattern against the input stream during the matching phase.

Next, we determine the cache misses that occurred in all the RE engines during the compilation and matching phase while processing the datasets. There are a number of factors responsible for slower program execution time and a cache miss is often considered one of those factors. The increase in complexity of RE pattern matching directly leads to more program instructions being generated during compilation. This degrades the ability to cache instructions in fast access memory leading to a higher cache miss rate and more instructions have to be fetched from the slower main memory. Overall, this reduces the performance of pattern matching. Depending on the application e.g. firewall, IDS this might cause some important steaming data to be dropped or clients experiencing a large response time as the data arrival rate exceeds the processing rate. As presented in Figure 4-10, both Regex-Processor and Rexg represented a high number of cache misses in their compilation phase as compared to the matching phase.

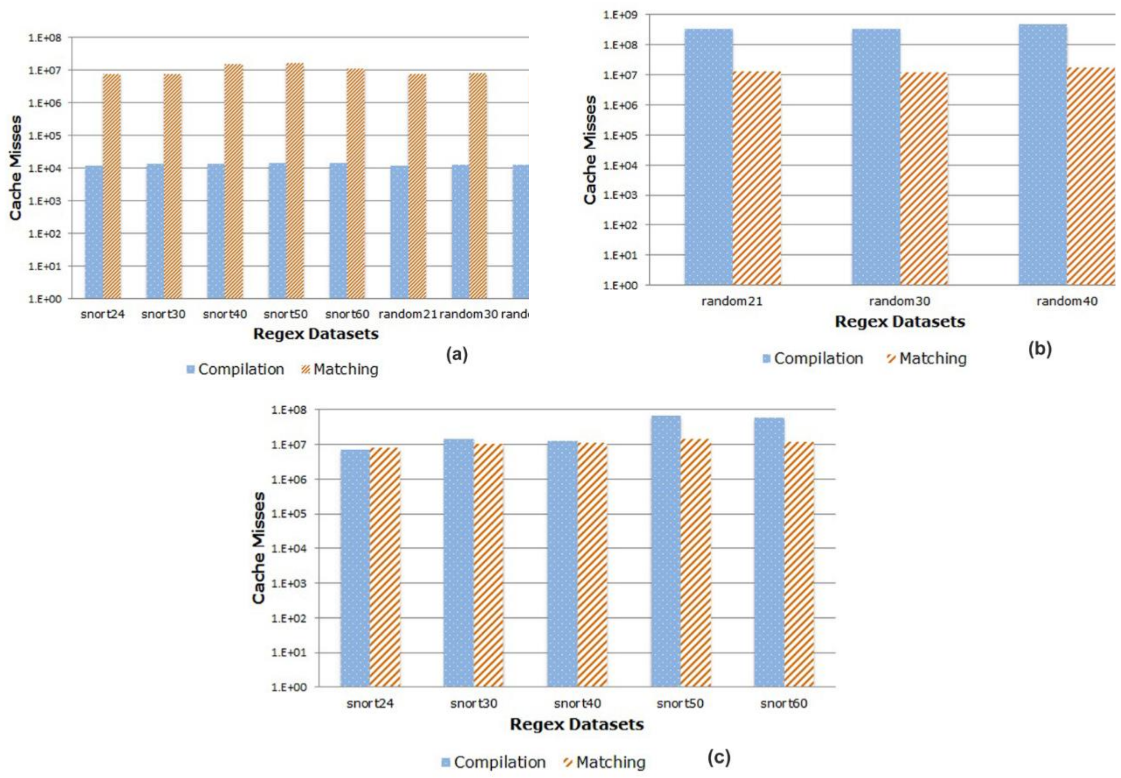


Figure 4-10 Cache misses in compilation and matching phase. (a)RE2 (b) Rexgrep (c) Regex-Processor.

The impact of cache misses on the execution time of both the phases can also be observed in Figure 4-9. The higher number of cache misses during the compilation of REs leads to higher execution time as well. Unlike Regex-Processor and Rexgrep, RE2 results in a small number of cache miss during compilation as compared to the matching phase. Figure 4-11 also demonstrates the cache misses per second for each of the RE engines. Here, both RE2 and Rexgrep results in a higher number of caches miss per second in the parsing phase as compared to their respective matching phases. Next, we determined Cache Misses to Instructions Ratio (CMIR) that calculates the number of cache misses per instruction. This ratio should be as low as possible. CMIR is quite low in all our experiments but if we compare this ratio between compilation and matching phases then the results depict that RE2 and Rexgrep have high CMIR in their compilation phase. In the case of Regex-Processor, CMIR is nearly the same for both the phases as represented in Figure 4-11.

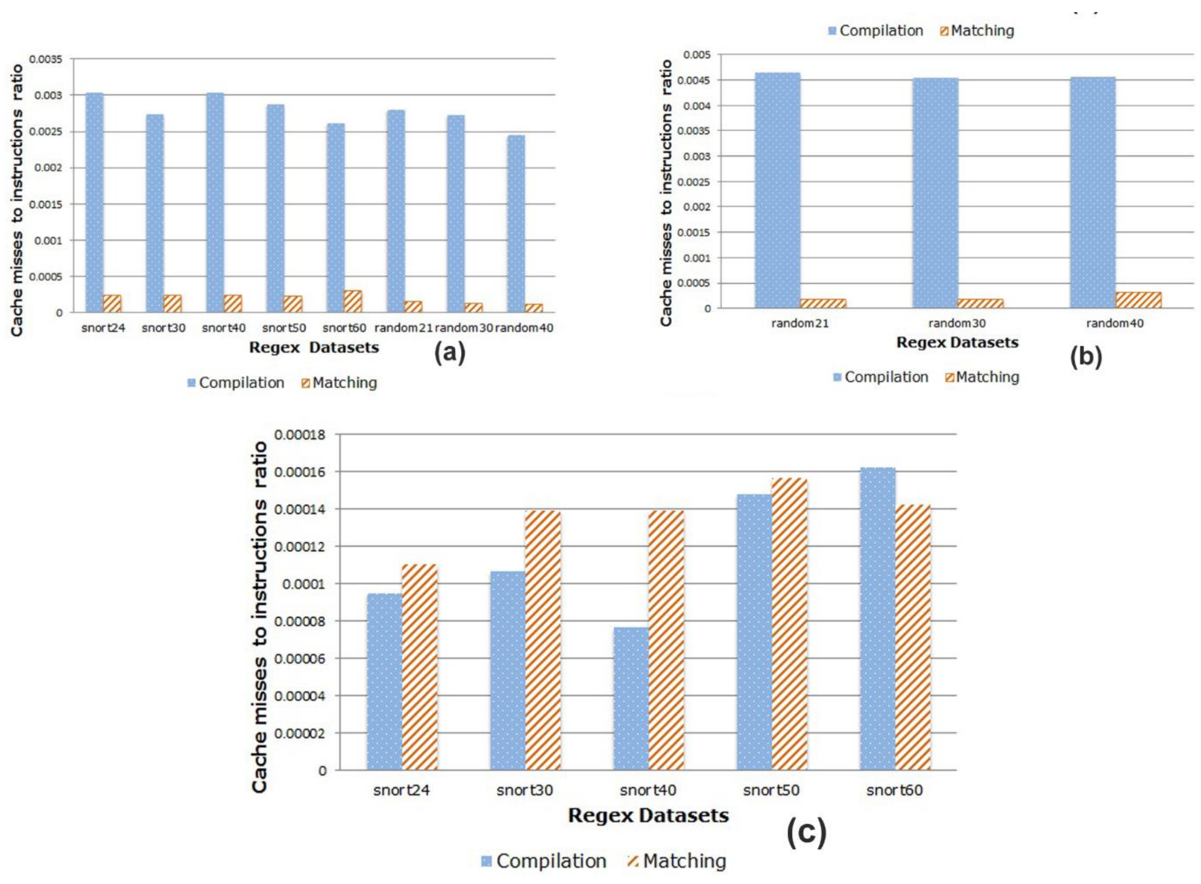


Figure 4-11 Cache misses to instructions ratio. (a)RE2 (b) Rexgrep (c) Regex-Processor.

The processing speed of the instructions is often highly affected by the use of complex instructions. Therefore, a comparison of the instruction set (determined using Intel's Pintool) is made that is used by the compilation and matching phases for all the three RE processing engines. The significant instructions are broadly classified into five categories: Arithmetic, Logical, Cmp and Test, Data Movement and Jump instructions as provided in Table 4-5. Due to space limitations, the results for Regex-Processor (snort50) and Rexgrep (random40) are only presented in Figure 4-12. It can be seen that the compilation phase has a higher set of instructions in all the categories as compared to the matching phase. In addition to this, data movement instructions are highly used during the compilation of patterns that often adversely affects the energy efficiency and performance of the system. Therefore, it can be concluded from these experiments that the compilation phase is highly memory intensive as compared to the matching phase.

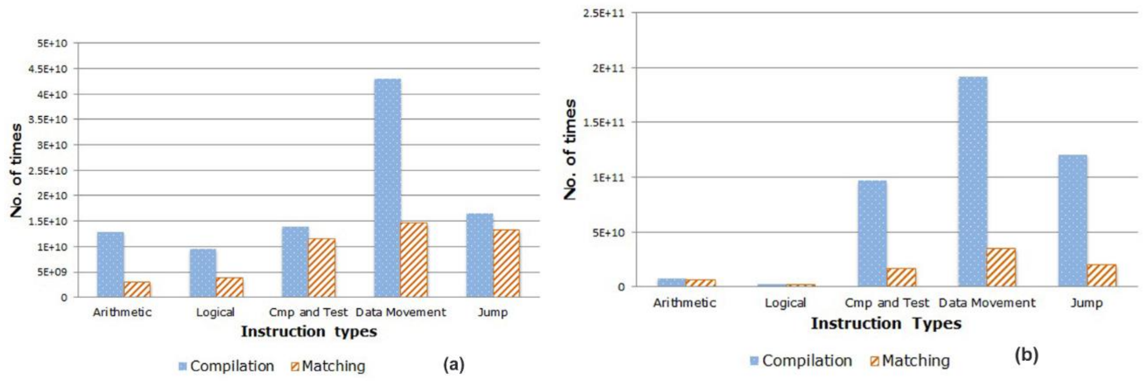


Figure 4-12 Occurrences of different instruction types in compilation and matching phase. (a)Rexgrep (random40), (b) Regex-Processor (snort50).

Table 4-6 Classification of opcodes into categories.

| Instruction Types | OPCODES |
|---|---|
| Arithmetic | ADD,DEC,INC,SUB,PSUBB, DIV,IDIV,IMUL,MUL,NEG |
| Logical | AND, OR, XOR, POR, PXOR, ROR |
| Cmp and Test | TEST, CMPXCHG, CMP |
| Data Movement | MOV,MOVD,MOVDQA,MOVDQU,MOVHPD, MOVLPD,MOVQ,MOVSQ,MOVSD XMM,MOVSX,MOVSXD,MOVZX |
| Jump | JB,JBE,JL,JLE,JMP,JNB,JNBE,JNL,JNLE,JNS,JNZ,JS,JZ |

### *4.4.2 Evaluation of JSON-based Patterns*

In this section, we evaluated the performance of our proposed JSON-based patterns with a particular focus on their compilation (or parsing) phase. Fig 4-13 presents an overview of the JSON pattern processing engine that is used for the evaluation. We used the Regex-Processor available at [42] to facilitate the compilation of JSON patterns. By default, Regex-Processor makes use of only REs to create automata that are further used for matching patterns against any input stream. However, we added another module in the Regex-Processor for processing JSON-based patterns. This module is marked as compilation phase-I in Figure 4-13. In addition to this, some modifications were also made in the Regex-Processor to make it compatible with JSON patterns. Moreover, we used five datasets of patterns, which includes snort 24, snort30, snort40, snort50 and snort60 for performance evaluation of JSON patterns.

Table 4-6 contains the opcodes used in pattern matching in the current research. These are the same datasets that we described in the previous chapter but there each dataset comprised of REs. Here, the patterns used in the dataset are the same but are expressed in JSON format. Once the pattern file is taken as input in the compilation phase-I then it is parsed using the Rapidjson library [34] to extract meaningful tokens from the file. Here, the term “meaningful tokens" refers to those tokens that play a critical role in the creation of FA. As a result, tokens like “StartofObject”, “EndofObject”, “Start of Array” and some Boolean values, etc. are ignored while parsing the pattern file (comprises of a number of JSON patterns).

The output of the compilation phase-I is a file of tokens. This file is then read in the next stage where the NFA is formed using the information conveyed by the token or a group of tokens. After the formation of the NFA, the engine makes use of its built-in algorithms to reduce the size of the NFA, which is then converted into the DFA. Next, the compiled DFA is used to match patterns against the input stream. We used different performance metrics for evaluation of the compilation phase. The metrics used for evaluation include cache misses, cache misses per second, cache miss to instructions ratio and time elapsed. The results are given in Figure 4-12 where it can be observed that the JSON approach has not resulted in a significant change over REs. In fact, both REs and JSON expressions have depicted nearly the same trend in performance.

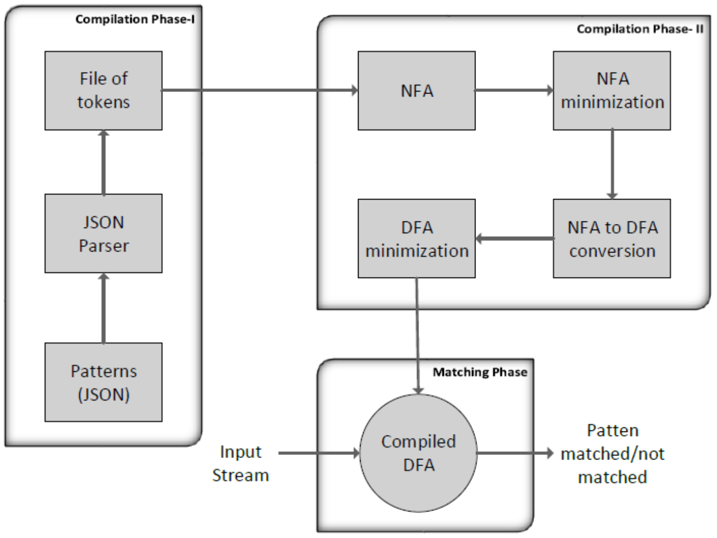


Figure 4-13 Overview of JSON pattern processing engine.

### *4.4.3 Evaluation of JSON-based patterns*

Figure 4-13 describes the overview of JSON pattern processing engine. For the empirical evaluation of the effectiveness of our work regarding the acceptance of syntax, we conducted a user study. On-line access to the JSON pattern processing engine and associated rule-sets was provided to a mix of programmers and students, the demographics summary of which is demonstrated in Table 4-7.

Table 4-7 Summary of demographics.

| Total: 26 | |
|---|---|
| Total Users | Postgraduate students: 15 (58%) Professionals: 11 (42%) |
| Age | 25-48 years |
| Working Experience in IoT | Yes: (27%) No: 0% |
| Experience in RE parsing techniques. | 3-6 years (19%) 8-12 years (23%) |

The procedure for the study was as follows:

1) The participants were initially provided the domain background regarding the use of pattern matching using REs without going into any details of the strengths or limitations of the current state of the art RE pattern languages.
2) The objective of the study was communicated, namely to determine the effectiveness of a new RE language for pattern matching.
3) The architectural design and side-by-side examples of expressions for the proposed and existing approaches for the snort dataset were provided.
4) Participants practiced writing their own expressions using online-access to the JSON pattern processing engine based on the syntax and usage details provided.
5) After completing the exercise, the participants answered the survey questions.

The survey results (Table 4-8) corroborate the expressiveness and extensibility of the proposed JSON-based RE pattern expression language.

### *4.4.4 Results Analysis*

This section summarizes the results and conclusions drawn from all of the above experiments. The experiments are divided into two sections where the first section signifies the critical nature of the compilation stage of the pattern matching engine. For unbiased evaluation, we performed all these experiments on three different RE processing engines which include RE2, Rexgrep, and Regex-Processor. Then, compilation and matching phases of these engines are compared on different metrics like time elapsed, cache misses, cache misses per second and CMIR. The datasets of patterns (REs) with varying complexities run against each matching engine. The results related to cache misses demonstrated that cache misses are relatively high in the compilation phase as compared to the matching phase.

Figure 4-14 illustrates the time elapsed and Figure 4-15 presents the cache misses on different dataset patterns. High cache misses and CMIR has a negative impact on a particular task as they are one of those factors that are often considered responsible for the slower execution of the task. In the present case, Regex-Processor and Rexgrep demonstrated higher execution time for the compilation phase that can be linked with higher cache misses. Furthermore, we also analyzed the instruction types involved in both the phases in order to determine the memory-intensive phase between the two. Again, it was observed that the compilation phase is much more memory intensive (a large number of data movement instructions) as compared to matching phase. Hence, the experiments conclude that the compilation phase of the patterns is also as significant as matching phase especially if patterns are compiled in real-time.

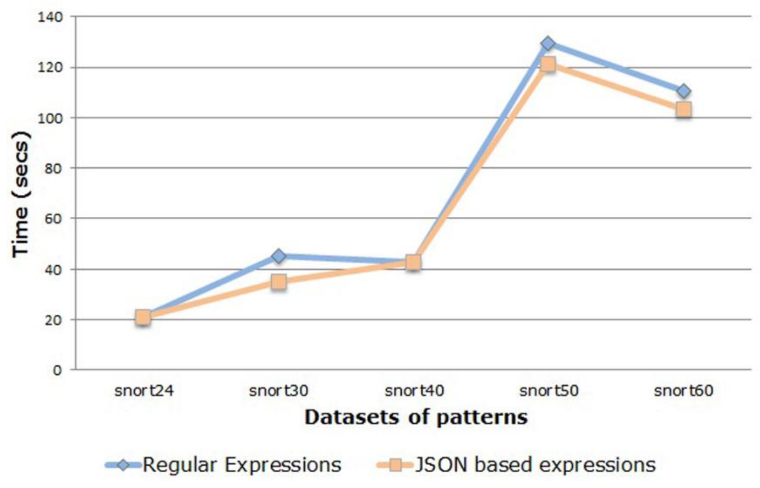


Figure 4-14 Time elapsed.

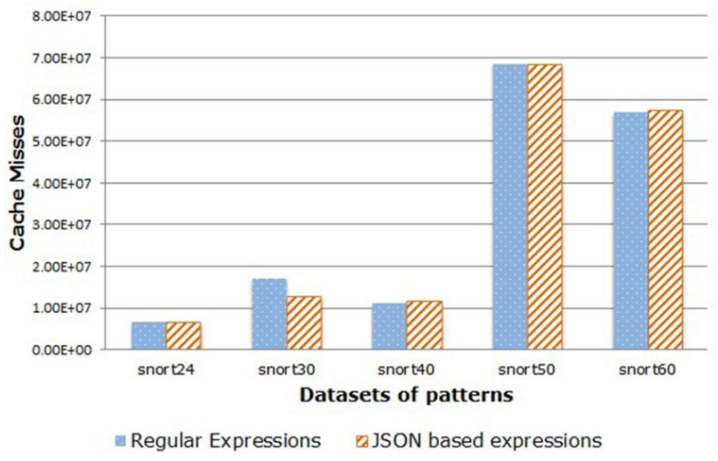


Figure 4-15 Cache misses.

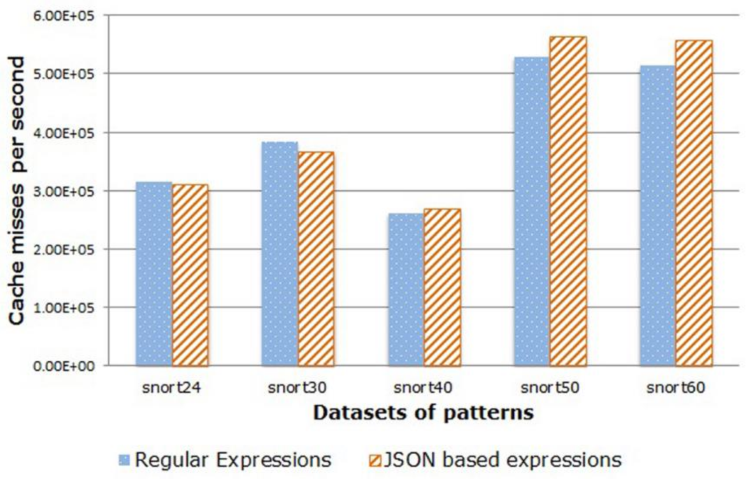


Figure 4-16 Cache misses per second.

The second part of this section evaluates the performance of our proposed JSON-based patterns. Again, JSON patterns are evaluated at the same time and cache related metrics. However, the comparison is made this time between JSON-based patterns and REs. JSON patterns and (optimized real-world) REs illustrated nearly the same results. This phenomenon demonstrates that JSON-based patterns can be used in matching engines without any efficiency compromise while providing an extensible and unambiguous notation. Figure 4-16 illustrates the cache misses per second on different datasets. Moreover, Figure 4-17 presents the cache misses to instructions ratio where the snort40 has the lowest and snort60 has the highest cache miss to instruction ratio.

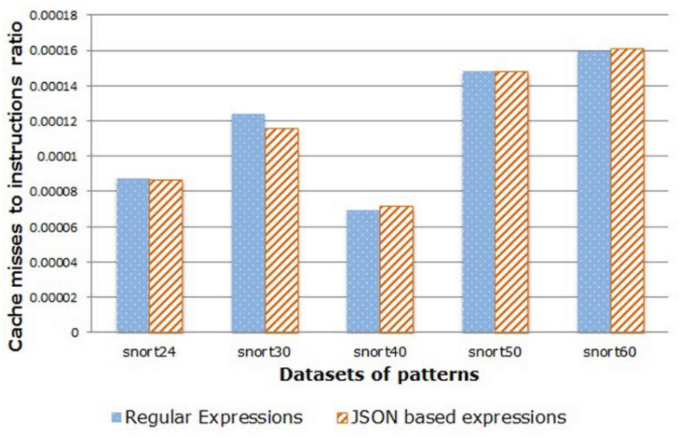


Figure 4-17 Cache Misses to instruction ratio.

The JSON-based pattern matching engine was also compared for studying accuracy with the three systems listed in the Table 4-4. All performed equally accurate for their supported RE datasets, as all worked on the principle of exact match. However, the coverage of each system is different as depicted in the table refrenced above. Only our presented system and RE2 were able to run all kind of regular expression sets. It is to highlight that due to the custom design, our developed system can also handle the approximate matching scenarios. Our research focus was to come up with a pattern matching engine with wider coverage, support for all matching scenarios and with efficient pattern compilation step, and the presented sytem ticks all the boxes.

Finally, the results of the user survey support the effectiveness of the approach. A majority of the users surveyed indicated that the proposed JSON-based RE language is extensible, expressive, user-friendly and useful for their development activities.

Table 4-8 Summary of survey questions and results.

| **Survey Questions** | **Results** | | |
|---|---|---|---|
| | **Yes** | **Some extent** | **No** |
| The JSON-based parser allows defining the patterns in a less ambiguous way as compared to other related languages. | 74% | 11% | 15% |
| The proposed JSON-based parser is extensible. | 86% | 7% | 7% |
| The proposed JSON-based parser is expressive. | 92% | 4% | 4% |
| I was easily able to learn the JSON-based RE syntax and define simple to complex patterns. | 69% | 19% | 12% |

| | | | |
|---|---|---|---|
| The JSON-based parser stimulated my interest in learning about good pattern writing techniques. | 58% | 15% | 27% |
| Were you already aware of the limitations of current RE-based pattern matching languages? | 31% | 38% | 31% |
| Application is user-friendly. | 46% | 23% | 31% |
| I would like to use the JSON-based parser in my development activities. | 65% | 19% | 16% |
| The abilities of the proposed parser are useful for my technical domain. | 74% | 12% | 14% |

## 4.5 Chapter Summary

In this research, we propose a JSON-based pattern matching engine as the first line of defense in order to solve the problem of LFA on the control channel. To the best of our knowledge, we are the first ones to propose the idea to protect the control channel against LFAs. Our work lies at the intersection of infrastructure security and enables virtualization. It aims at securing large-scale infrastructures from LFA and providing seamless operation during the attacks. In this chapter, we presented high performing co-processing architecture which performs efficient pattern classification using the DFA techniques. It stores a repository of the attack patterns however it is equally feasible in using the novel traffic patterns which are aimed by the smart adversaries to deceive the pattern matching engines. Current state-of-the-art techniques employ pattern matching techniques to identify malicious traffic patterns. The patterns of the malicious traffic are identified and the incoming traffic is compared with those patterns. The requirements for the DCI technologies need faster algorithms to perform the rea-time packet inspection. More often the network security applications are implemented in the high-speed networks as middle boxes hence the algorithms need to perform the tasks in near real-time. The DCI techniques are often deployed as hardware solutions which possess a very high-cost memory; therefore, space efficiency is required in the DCI solutions. Our proposed DCI engine consumes a low amount of space and is extremely time-efficient. It offers efficient time-space efficiency which makes it an optimal solution for pattern matching problems.

We have developed a pattern matching engine which incorporates a repository of malicious traffic patterns. Some attackers confuse the DCI engine by employing unconventional fragmented IP packets however, our DCI engine can detect these novel type of attack patterns. Moreover, the processor in the prevailing solutions may not be equally supportive in novel traffic patterns however, the processor we used can detect the new attacks efficiently.

We have discussed the importance of pattern matching technique in detecting LFA. Pattern matching provides an efficient mechanism to represent different data patterns using REs. In the same way, malicious network traffic patterns can be represented using REs. Hence we can learn the flood traffic patterns and compare them with the attack traffic and mitigate them. We use pattern matching as the first line of defense against LFA. In the next chapter, we discuss ML techniques to detect and defend against LFA, tools used in this context, and algorithms which can provide efficient results in classifying network traffic.

# CHAPTER 5. MACHINE LEARNING TOOLS, TECHNIQUES, AND ALGORITHMS

In this Chapter, we aim to answer research questions 7 & 8 given in the section 1.7.1. It is important to present a study of various ML algorithms relevant to traffic classification, and to explore the various training datasets used by researchers globally. The intent of the upcoming sections is to provide basis for designing a ML classifier capable to use multiple data sets, and then conduct some basic experimentation to ascertain the suitability of it as a module in the larger CyberPulse solution. Therefore, the study of different ML algorithms, and training these through various tools with variety of dataset is essential to shortlist the tools that we will be using to train and test CyberPulse traffic classifier. Towards this end we also provide an overview of selecting the right tools for ML problems by considering multiple performance parameters. To help reader understand the context, we also discuss the steps to select the right ML algorithm depending upon the problem scenario. Finally, we connect all the dots and provide a detailed explanation on the training of CyberPulse ML classifier including the preprocessing, the dataset explanation, and the steps for training the classifier.

ML algorithms are being deployed in every field of life during the past few years. Previously ML was supped to be used in the traditional computing studies only, however, due to the huge amount of data production during recent years, they are being used in every field of life [220]. Modern networks also produce a huge amount of data on traffic forwarding, network attacks, surveillance, and traffic engineering. This data can be utilized to perform security checks in the networks. Efficient handling of data facilitates the decision making which ultimately helps current data center networks to operate without facing network outages.

In this chapter, we provide a detailed discussion on ML algorithm, tools, their working principles, selection of the relevant tools, and algorithms for the required problems. This chapter will facilitate the network security providers to select appropriate tools and algorithms for their problems and will make the basis of the key decisions taken in the CyberPulse prototype.

The ML algorithms provide the ability to learn from the data and to aid in decision making. The learning in the ML algorithms can be regarded as learning from the functions or leaning the intrinsic structures in the unlabeled data. Moreover, an instance-based learning example can be employed to learn from the training data which can produce the class label for the new record without much human intervention. This is performed by comparing the new record with the already available records. There are a huge number of algorithms available which produce results based on intrinsic computational methods to transform the inputs into the output. Selection of right ML algorithm poses a vital challenge on the ML experts who need to solve a complex computation problem. As different algorithms are used for disparate problems as classification, clustering, anomaly detection, outlier detection, variance detection, and outcome prediction algorithms are used for different types of problems.

For an optimum decision on ML algorithm selection, different algorithms can be analyzed by running a small-scale experiment and finally the best performing algorithm can be chosen. Keeping in view this challenge in the upcoming section, we provide a detailed analysis of the ML algorithms which highlight the core properties of the algorithms for specific tasks. Below is a list of well-known algorithms. Details of each are intentionally being skipped as any ML text contains step by step guides on these and the variants.

- K-means Clustering
- Principal Component Analysis
- Logistic Regression
- Support Vector Machines
- Feed Forward Neural Networks
- Conditional Random Fields
- Decision Trees
- Artificial Neural Networks

Some experimentation in the forthcoming section 5.4 and 5.5 cites these algorithms for model training and tuning.

## 5.1 Selection of Machine Learning Tools

ML tools are an important part while solving complex real-world problems where the selection of the right tool can become an important aspect in working with the ML algorithms. Proper tools selection assists in automating the process of applied ML. Without a properly selected tool, each capability needs to be developed manually from scratch which consumes a lot of time. The tools in this domain make it easy to perceive an idea and quickly getting the results out of it, but there are plenty of free ML tools available on the internet. Therefore, a short survey on tooling is deemed necessary here.

Flood traffic classification comes under the category of normal data classification which do not require support for big data. Considering this, Weka seemed to be a logical choice to perform the classification of the given traffic dataset. However, for the benefit of the research community, a comparative analysis was intended to assess the feasibility and various features of all useful freely available tools.

Table 5-1, compares state-of-the-art ML tools which include Weka, D4J (Deep Learning4J), Tensor Flows and Encog3. Selection of these tools is based on their extensive use for current ML problems. The comparison is performed based on parameters, including source code, help and support, license type, programming language support, compatibility, documentation, and performance. Figure 5-1 describes the parameters for the programming tool selection for ML tasks. The detailed description of comparison parameters is given in the subsections.

Table 5-1 Comparison of deep learning tools. (scale 0 – 10)

| **Comparison Parameters** | **Weka** | **D4j** | **TensorFlow** | **Encog3** | **PyTorch** |
|---|---|---|---|---|---|
| **Source Code** | Java framework | Java framework | Python API over C++ engine | C# framework | Lua-based |
| **License** | Open source | Open source | Open source | Open source | Open source |
| **Programming Language** | Java | Java | Python | C# | Python |
| **Compatibility** | No issues | No issues | Less backward compatibility | Less backward compatibility | Less backward compatibility |
| **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 & Support** | 9 | 6 | 7 | 5 | 4 |
| **Documentation** | 9 | 6 | 7 | 5 | 6 |
| **Performance** | 9 | 7 | 6 | 8 | 9 |

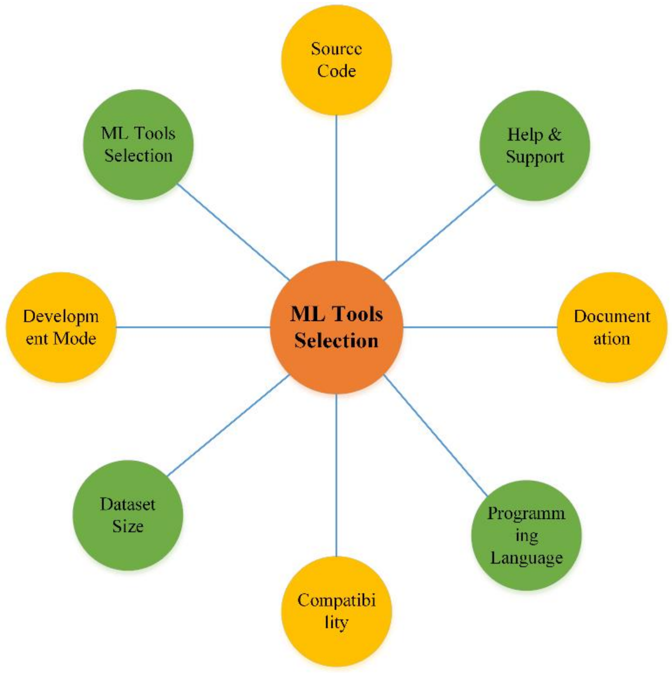


Figure 5-1 Programming tools selection criteria.

From the table above PyTorch and Weka are the two tools with a very high rating on performance scale. At this stage it is important to discuss each of the five mentioned tools one by one.

### *5.1.1 WEKA*

WEKA facilitates the core of ML problems, i.e. classification and clustering. It contains useful preliminary features to do data-related tasks including filtering, preprocessing, classification, aggregation, clustering, and many others. It facilitates the development of novel approaches in the field of ML. It has been developed using JAVA programming language. WEKA offers SQL connectivity using JAVA Database Connectivity (JDBC). It offers an explorer interface moreover it also offers the same functionality using the knowledge flow and Command Line Interface (CLI).

### *5.1.2 Encog3*

Encog3 is a C# based framework for ML tasks, it can be used for all the ML problems along with a range of medium-sized datasets. However, there is limited support available for Encog3 and it requires strong skills of C# language to develop ML solutions. It is mainly used for neural network programming. The weight, thresholds and the number of hidden layers in ANN can be efficiently computed using Encog3.

### *5.1.3 TensorFlow*

Google TensorFlow is the most widely used deep learning library in the world. Google uses TensorFlow to classify the datasets and provide the user with a good experience. TensorFlow is a state-of-the-art ML tool which supports Big Data, however, TensorFlow is fully supported only by python language. It is an advanced ML repository for solving ML problems which is suitable for big datasets, the computations performed by TensorFlow are highly accurate, but it requires slightly higher time for model development as TensorFlow operates based on graphs.

### *5.1.4 Deeplearning4j*

Deeplearning4j ML tool is based on the java framework and requires higher performance computing infrastructure for operation. It supports the state-of-the-art Big Data frameworks like Apache Spark, and Hadoop to boost the model training. Moreover, it uses open-source libraries and its backend language is Java which enables it to be utilized with any JVM language like Scala, Clojure, and Kotlin. The Eclipse D4j is the first of its kind library to be employed with Java and Scala. It introduces deep neural networks from the shallow nets which individually is called as a layer enabling the data scientists to employ multiple autoencoders like recurrent and conventional nets.

### *5.1.5 PyTorch*

It is a deep learning framework which is based on Lua. The underlying operation of PyTorch is based on TensorFlow which works by generating Directed Acyclic Graph (DAG) of each model. A graph is defined prior to the model generation in TensorFlow and the input is generated from the *tf.Session* and *tf.Placeholder* outer world interfaces. Alternatively, PyTorch works in a dynamic manner where the nodes of the DAG are dynamically defined and executed. Moreover, session and placeholder variables are not required. This framework is coupled to use with python language providing seamless integration. PyTorch can perform scalable ML operations in a distributed way where a rich set of tools and libraries extends its support to be used for compute-intensive ML tasks including Computer Vision (CV), Natural Language Processing (NLP) and various other.

Based on this discussion, it can be ascertained that Weka is suitable for small datasets classification and clustering tasks. Encog3 can be utilized ML tasks on medium-sized datasets. D4j and TensorFlow can be deployed for big datasets and complex ML tasks. For basic experimentation we used Weka, while for extensive evaluation and detailed modeling we benefitted from PyTorch. The details are given in the Chapter 6.

## 5.2 Selection of Classification Datasets, Features and ML algorithm

Recently, ML techniques for network traffic classification has gained immense popularity where multiple authors worked on the traffic classification using different traffic features. Table 5-2 and Table 5-3 contain different research paper titles, their technique name and the network statistics/features used for traffic classification.

Table 5-2 Statistics used in the previous research papers.

| Sr. | Paper Title | Technique Name | Statistics Used |
|---|---|---|---|
| 1 | Towards Autonomic DDoS Mitigation using SDN | Distributive Collaborative Framework | “IP Source, IP destination, Port Source, Port Destination” |
| 2. | Targets Can Be Baits: Mitigating Link-flooding Attacks with Active Link Obfuscation | Linkbait | Bot detection rate, (identified/total bots), False Positive Rate ( legitimate hosts wrongly identified as bots/ total legitimate hosts) |
| 3. | SDN-Guard: DoS Attacks Mitigation in SDN Networks | SDN Guard | “Flow throughput, Switch TCAM usage, Link bandwidth usage” |
| 4. | Mitigating Crossfire Attacks using SDN-based Moving Target Defense | MTD | Average Link Usage, End to End Delay |
| 5. | Towards Detecting Target LFA | LinkScope | Packet Loss Rate, Round trip time, Jitter, Number of loss pairs, Available bandwidth, Packet rerouting, Connection failure rate |
| 6. | An Efficient DDOS Detection with Bloom Filter in SDN | Bloom filter | packetCount, byteCount durationSeconds |
| 7. | NSL-KDD Dataset for Training Model | ML | Duration, Protocol_type, Service, Src_bytes, Dst_bytesCount, Srv_count (no. of connections to the same service) |

The paper # 7 listed in the table above, helped us find out the statistics set to start with. The Distributed-SOM and DyProSD papers listed in the Table 5-3 below helped us figure out the features to note from the dataset, and made the basis for our ML module of CyberPulse.

Table 5-3 Flooding Attack Datasets used in literature.

| Paper Title | Year | Dataset Used | Technique | Features | Access |
|---|---|---|---|---|---|
| Distributed-SOM | 2017 | Anonymized Internet Traces DoS Attacks, CAIDA | Distributed Self Organizing Maps | Number of Flows Number of Packers per Flow, Number of bytes per flow, Duration | Anonymized Internet Traces |
| Adaptive artificial immune networks | 2017 | Anonymized Internet Traces, DoS Attacks, KDD, CAIDA | NA | NA | Anonymized Internet Traces, CAIDA, KDD |
| DyProSD | 2016 | Anonymized Internet Traces, DoS Attacks, CAIDA, MIT Lincoln Laboratory Datasets 1999 | C4.5 Naïve Bayes Decision Tree | Source IP, Destination IP Sampled Interval time Flow ID, Total Number of connections | Anonymized Internet Traces, MIT Lincoln |
| OpenFlowSIA | 2016 | Anonymized Internet Traces DoS Attacks, CAIDA | 1.SVM | Protocol, IP Source, IP Destination, Port Source, Port Destination | Anonymized Internet Traces, CAIDA |
| A lightweight DDoS | 2015 | CAIDA, Anonymized Internet Traces DoS Attacks | Hop Count Filter Algorithm | Source IP, Destination IP | Anonymized Internet Traces, CAIDA |
| TDFA | 2014 | Anonymized Internet Traces DoS Attacks, CAIDA | IP Trace-back algorithm | | Anonymized Internet Traces, CAIDA |

Through this section, we aim to select an appropriate dataset for traffic classification to train CyberPulse classifier. Selecting a relevant dataset for the classification task is a key to get better results. Each problem requires extensive analysis and modeling according to its domain which needs different preprocessing and data filtering steps.

Table 5-4 demonstrates flooding attack datasets in the literature used for the classification of the attack traffic. After selection of a suitable dataset, the next step is to select the features that will be used for the classification. Regression, classification, anomaly detection, and predication are some of the problem types that one need to carefully review while selecting a dataset. Another key element is to find out the number of features available in the given problem domain and the data types these features belong to.  Moreover, the performance baseline required from the ML problem and knowing the best- and worst-case scenarios should also be carefully considered.

Table 5-4 Important Datasets used for Flooding Attacks with their links.

| Sr. No | Dataset Repository |
|---|---|
| 1. | MIT Lincoln Laboratory LLSDDoS Dataset 199 |
| 2. | KDD cup Dataset 1999 |
| 3. | UCLA Dataset 2001 |
| 4. | CAIDA DDoS Attack Dataset 2007 |
| 5. | DARPA DDoS attack dataset 2009 |
| 6. | TUIDS DDoS Dataset 2012 |
| 7. | Booter DNS Dataset 2014 |

After a careful review of the datasets based on the described guidelines, we selected the data sets given at Sr.No 3, and 5.

### *5.2.1 ML Algorithm Selection*

There are many ML algorithms available to choose from when we come across a ML problem. The selection of the algorithm depends on the type of problem we are dealing with; however, the performance of multiple ML algorithms depends on the size and the structure of data. So, plain trial and run methods can be employed for the selection of an appropriate algorithm.

However, pros and cons are associated with each ML algorithm, Figure 5-2 demonstrates the classification algorithm selection strategy. Initially, when a ML task is assigned, the first question to ask is that how many numbers of classes are to be predicted, if there are two classes, then further decision depends on multiple other aspects i.e. accuracy, training time, and performance. In the next step, analysis is performed to assess accuracy and training time.

In CyberPulse, we initially selected ANN technique since our main concern was accuracy as the system accuracy highly depends on the false positive rate.

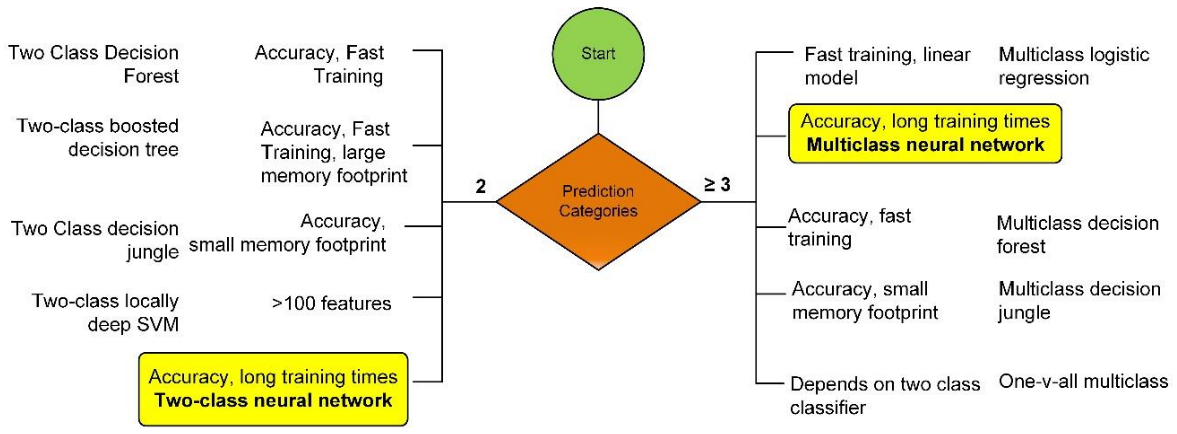

Start
Prediction Categories
2
≥ 3
Two Class Decision Forest
Accuracy, Fast Training
Two-class boosted decision tree
Accuracy, Fast Training, large memory footprint
Two Class decision jungle
Accuracy, small memory footprint
Two-class locally deep SVM
>100 features
Accuracy, long training times
Two-class neural network
Fast training, linear model
Multiclass logistic regression
Accuracy, long training times
Multiclass neural network
Accuracy, fast training
Multiclass decision forest
Accuracy, small memory footprint
Multiclass decision jungle
Depends on two class classifier
One-v-all multiclass

Figure 5-2 ML algorithm selection criteria.

### *5.2.2 Training of CyberPulse Traffic Classifier using Multiple Algorithms*

For initial training and testing of CyberPulse classifier, a ML model was trained using 10 different algorithms for LFA classification. The training dataset called, Burst Header Packet (BHP) flooding attack, was downloaded from the UCI ML repository. Python 3.6 programming language with Anaconda and Jupyter Notebook as an integrated development environment was used. The training dataset contained 1075 records and 22 features. The first four instances of the dataset are presented in Table 5-5, the outcome was very encouraging.

Table 5-5 Training dataset introduction.

| Node | UBR | PDR | Bandwidth | PLR % | LBR % | PRR % | Used Bandwidth |
|---|---|---|---|---|---|---|---|
| 3 | 0.82 | 0.19 | 1000 | 19.03 | 19.03 | 0.81 | 822.04 |
| 9 | 0.27 | 0.73 | 100 | 72.89 | 72.91 | 0.27 | 27.55 |
| 3 | 0.92 | 0.09 | 900 | 9.03 | 9.03 | 0.91 | 831.34 |
| 3 | 0.37 | 0.64 | 100 | 63.74 | 63.77 | 0.36 | 36.88 |

## 5.3 Data Preprocessing

This section gives details on dataset pre-processing, class and log transformation and feature normalization etc for the CyberPulse ML traffic classifier. The UCI dataset was loaded in the Jupyter Notebook and *expert analysis* method was applied to summarize the features. Categorical variables were excluded before the analysis because they belong to limited categories and certain mathematical operations couldn't be performed. Data description illustrates the statistics of all the attributes including, count, mean, standard deviation, minimum, maximum, 25%, 50% and 75% of the data as presented in Table. 5-6.

Table 5-6 Expert analysis of the training dataset.

| Statistic | UBR | PDR | Bandwidth | PLR % | LBR % | PRR% | Used Bandwidth |
|---|---|---|---|---|---|---|---|
| Mean | 0.59 | 0.41 | 540 | 41.16 | 41.19 | 0.59 | 340.78 |
| Std | 0.19 | 0.18 | 289 | 18.35 | 18.37 | 0.18 | 232.14 |
| Min | 0.24 | 0.09 | 100 | 8.61 | 8.61 | 0.23 | 27.55 |
| 25% | 0.45 | 0.25 | 300 | 24.75 | 24.75 | 0.43 | 138.40 |
| 50% | 0.58 | 0.44 | 500 | 43.80 | 43.80 | 0.56 | 291.59 |
| 75% | 0.76 | 0.56 | 800 | 56.67 | 56.67 | 0.75 | 515.18 |
| max | 0.93 | 0.77 | 1000 | 76.79 | 76.79 | 0.91 | 867.03 |

The *packet lost rate* feature contained some missing values which were replaced with a median value of the same feature to ensure justifiable distribution of all the features.

### 5.3.1 *Class Transformation*

The actual class distribution in the BHP flooding attack training dataset was Block, NB-No Block, No Block, and NB-Wait. We transformed the classes into Flooding and legitimate, where Block corresponds to flooding and the rest of the classes assigned to legitimate. The final class distribution can be observed in Figure 5-3. Here 88% of the values represent legitimate and 12% are related to flooding class.

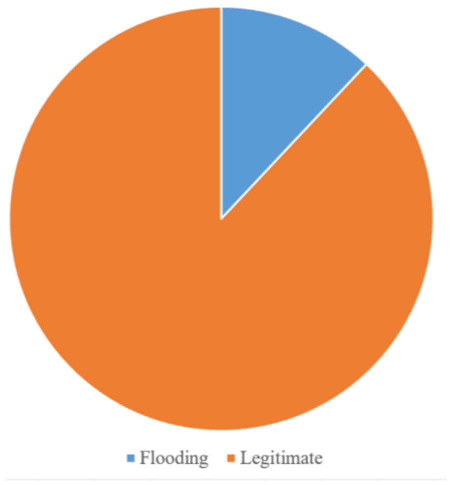


Figure 5-3 Class distribution of BHP flooding attack dataset.

### 5.3.2 *Log Transformation of Skewed features*

Data visualization demonstrated the following features were skewed.

- Used Bandwidth
- Lost Bandwidth
- Packet Received
- Received Byte
- Flood Status

In order to remove this skewness, python's *np.log* function was applied, subsequent to application of the log function, these attributes were plotted again. The results of the plot are provided in Figure 5-4.

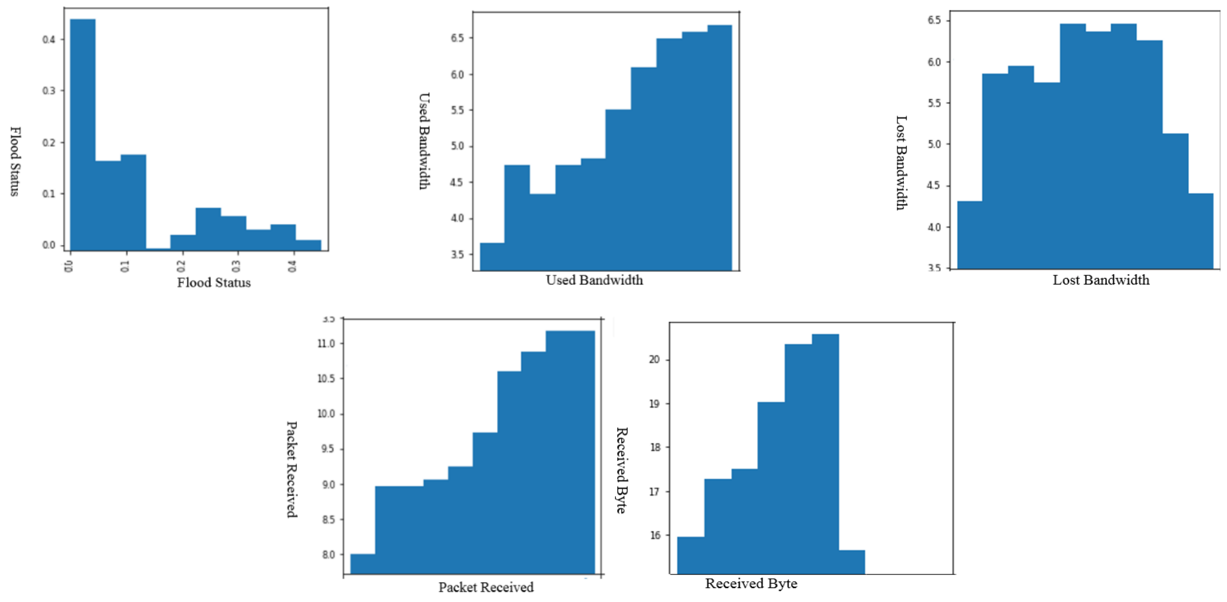


Figure 5-4 Illustrates features distribution after removal of the skewness.

### *5.3.3 Feature Normalization*

In this step, scaling was used to standardize the range of independent features. The features were rescaled in a way that they attained the characteristics of a standard normal distribution having a mean value of zero and standard deviation of one. Feature scaling has a high impact on the results as our training dataset contains features which consist of a diverse range, units, and magnitudes. In this regard, *Mini-Max* scaling was performed on the dataset which transformed the values in the range of 0 to 1 Table 5-7 demonstrates the dataset after the normalization operation.

Table 5-7 Dataset state after applying data normalization.

| Node | UBR | PDR | Bandwidth | PLR % | LBR % | PRR | Used Bandwidth |
|---|---|---|---|---|---|---|---|
| 3 | 0.85 | 0.15 | 1.00 | 0.15 | 0.15 | 0.85 | 0.98 |
| 9 | 0.058 | 0.94 | 0.00 | 0.94 | 0.94 | 0.06 | 0.00 |
| 3 | 0.99 | 0.0062 | 0.89 | 0.06 | 0.01 | 0.99 | 0.99 |
| 3 | 0.19 | 0.81 | 0.00 | 0.81 | 0.81 | 0.19 | 0.083 |

### 5.3.4 *Feature Selection*

A Recursive Feature Elimination (RFE) technique was used for feature selection which considers wrapper method built on top of various other algorithms i.e. SVM or regression. This helped in model development based on different data subsets. The RFE repeatedly construct a model and choose from the best performing features.

### 5.3.5 *Shuffle and Split Data*

The data was split into training and test set in the form of an array of features where test size was set to 0.2. Figure 5-5 presents the overview of the dataset.

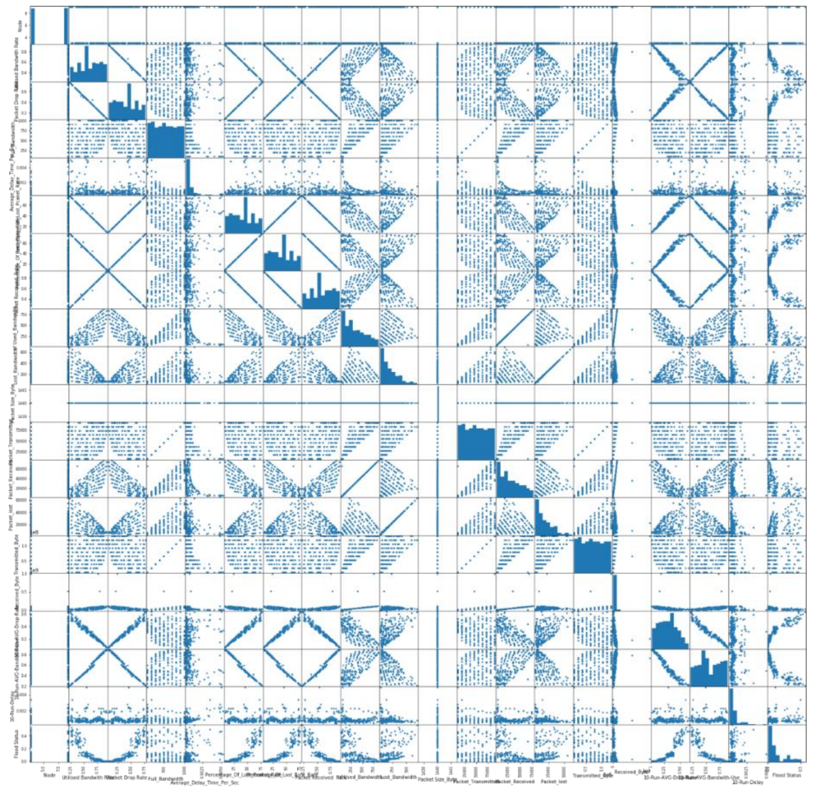


Figure 5-5 Overview of the dataset.

The parameters of all the following algorithms were defined, and then all the models were initialized.

1. Support Vector Clustering
2. K-Nearest Neighbor
3. Decision Tree
4. Logistic Regression
5. Random Forest
6. Adaptive Boosting
7. Stochastic Gradient Decision
8. Bagging Classifier
9. Naïve Bayes
10. Multi-Layer Perception

The results of cross-validation are then printed, the model is fit to the training data using slicing with sample size. Input parameters applied to the models are given in Table 5-8.

Table 5-8 Inputs parameters applied to the model.

| Input | Detail |
|---|---|
| Learner | The learning algorithm to be trained |
| X_train | The features training set |
| y_train | The training set |
| X_test | Features testing set |
| y_test | The testing set |

## 5.4 Model Training & Hyper-Parameter Tuning

In this section, we discuss the results of the training and testing of all the implemented ML algorithms. The evaluation metrics were $F_1$Score and Accuracy, where $F_1$Score corresponds to the weighted harmonic mean of precision and recall. A $F_1$Score of 1 is the best performing value, and a score of 0 corresponds to the worst-case scenario. The accuracy score is the ratio between correct predictions divided by the total number of predictions. Similarly, the Receiver Operating Characteristic (ROC) curve and Learning Curve were plotted for the evaluation of each algorithm. ROC curve is drawn to illustrate the relationship between true positive and false positive values, which represents the accuracy of class discrimination in a binary classifier. Table 5-9 summarizes the results of the training of the dataset.

The Area Under Curve (AUC) is important in ROC curve, where AUC value of 1, represents the perfect separation of the classes. Learning curve describes the cross-validation score and training score of a model on a varying number of training samples. It demonstrates the behavior of a model on adding more training instances if the training score and cross-validation score converges on a small training sample then adding more data will not benefit the model. The accuracy equation can be computed by equation (5-18) and the $F_1$Score can be computed by equation (5-19).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (5\text{-}1)$$

$$F1Score = 2 * \frac{precision * recall}{precision+recall} \quad (5\text{-}2)$$

Table 5-9 Evaluation metrics of all the algorithms.

| **Algorithms** | F$_1$Score | | **Accuracy** | |
|---|---|---|---|---|
| | **Training** | **Testing** | **Training** | **Testing** |
| Ad boost | 1 | 1 | 1 | 1 |
| BC | 0.98 | 0.96 | 0.98 | 0.96 |
| DT | 1 | 1 | 1 | 1 |
| KNN | 1 | 0.99 | 1 | 0.99 |
| LR | 0.98 | 0.95 | 0.98 | 0.95 |
| MLP | 0.99 | 0.98 | 0.99 | 0.98 |
| NB | 0.97 | 0.98 | 0.88 | 0.93 |
| RF | 1 | 1 | 1 | 1 |
| SGD | 1 | 1 | 1 | 1 |
| SVC | 0.94 | 0.91 | 0.82 | 0.89 |

### *5.4.1 Adaptive Boost and Bagging Classifier*

Adaptive boosting is a ML algorithm which utilizes ensemble learning method to develop a precise learning algorithm. Initially, it chooses a baseline algorithm e.g. Naïve Bayes and iteratively, increase the its performance by taking into consideration the incorrectly classified instances [221]. In adaptive boost algorithm, the evaluation metrics i.e. F$_1$Score and Accuracy for training and testing were 1.0 which correspond to the excellent performance of the trained model. The learning curve plot demonstrates that the cross-validation and training score converge on a sufficient amount of training samples.

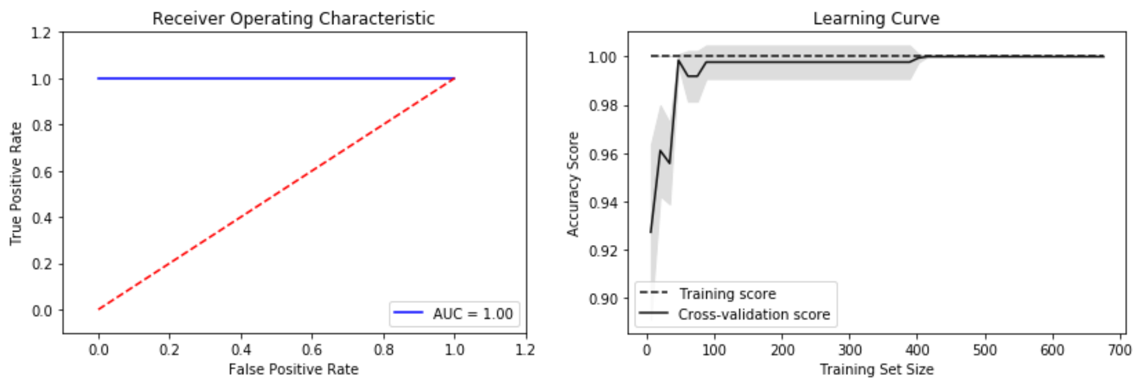


Figure 5-6 Learning curve converges at higher training instance and a perfect AUC score provides excellent separation of the classes.

The ROC curve given in Figure 5-6 represents excellent results where the area under the curve is equal to 1, which is the perfect separation of the flooding and legitimate classes.

Bagging Classifier (BC) is one of the bootstrap methods which invites individuals for its ensemble by training each of the classifiers. It performs this operation by randomly redistributing the instances of the training set [222]. Results of the BC can be observed in Figure 5-7 and Table 5-9. The Learning Curve converge at lower training instances which demonstrates that adding more training instances will not benefit the trained model. The AUC value on ROC curve corresponds to 0.83 which describes a fair separation of the predicted classes.

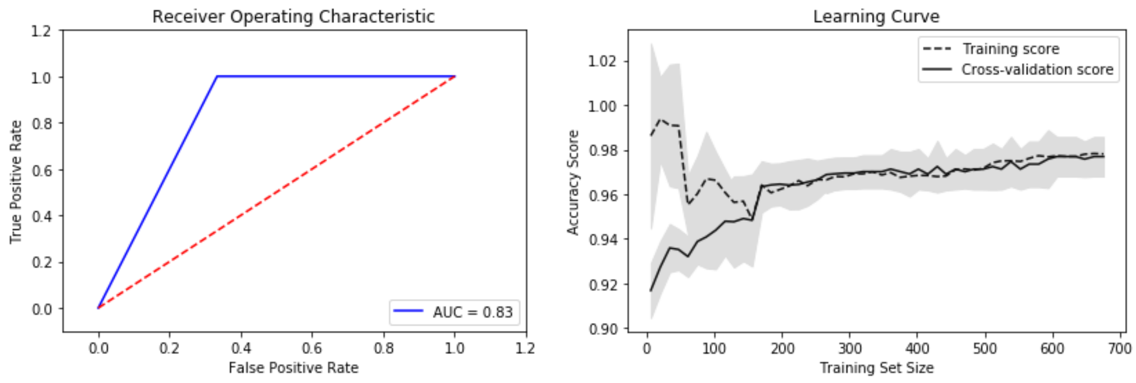


Figure 5-7 Learning curve converged at 150 training instances and AUC value of 0.83 on ROC Curve is observed.

### *5.4.2 Decision Tree and K-nearest Neighbors*

Decision tree is a non-parametric ML modeling technique, used for classification and regression problems. A decision tree provides hierarchical and sequential decisions about the outcome class on the basis of training data [223]. The ROC curve demonstrates excellent results where AUC is 1, which is a perfect separation of the flooding and legitimate classes. The performance evaluation values of F1Score and Accuracy corresponded to 1 which represents the best performance of the model.

The learning curve in Figure 5-8 expresses the results of cross-validation and training score. Both curves converge at lower training instances which describes that adding more training instances will not benefit the trained model.

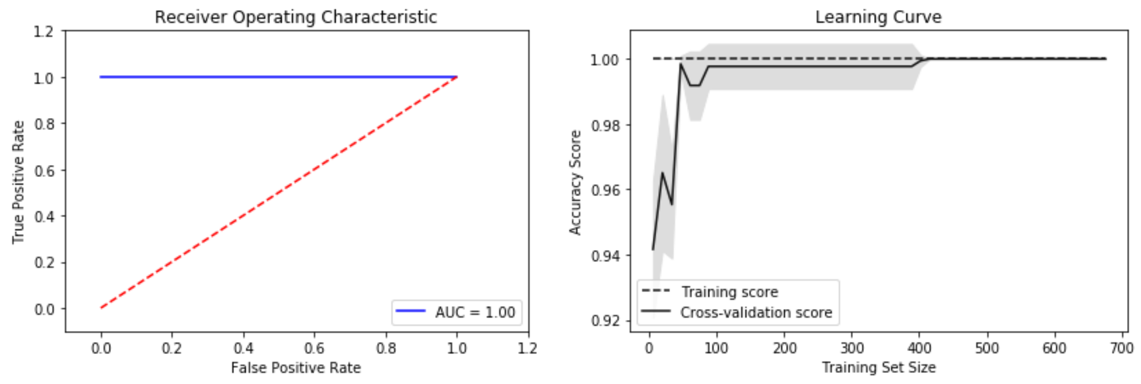


Figure 5-8 A perfect AUC score on ROC while Learning Curve converges around 400 training instances.

K-nearest Neighbors is one of the most used classification algorithms which predicts the likelihood of a data point to be a member of a group on the basis of nearest data points [224]. KNN also provided the best performance fit where values of evaluation metrics were 1.

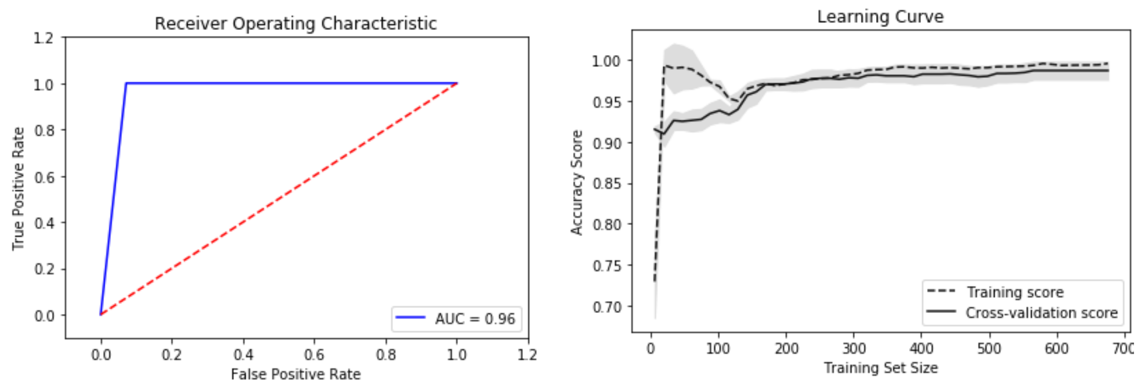


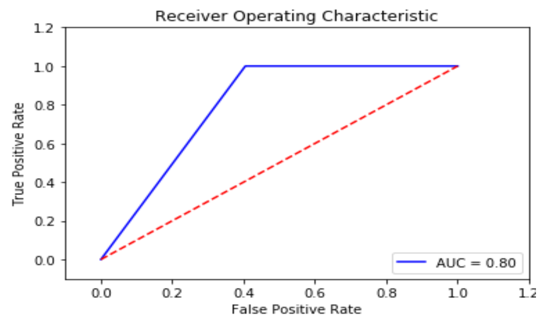
Figure 5-9 An excellent ROC score which corresponds to a perfect separation of the final classes, the Learning curve illustrates that adding more number of instances will not benefit the model.

Figure 5-9 represents the ROC and Learning Curve of the KNN model. The learning curve converges around 200 instances of the training set. The ROC curve illustrates excellent results where the area under the curve is 0.96, which expresses a perfect separation of the flooding and legitimate classes.

### *5.4.3 Logistic Regression and Multi-Layer Perception*

Logistics regression predicts the probability of an outcome that will only result in a binary value. It generates a logistic curve limited by binary values [225]. The results of the LR classifier are given in Table 5-9 and Figure 5-10. Evaluation metrics of F1Score and

Accuracy for the testing set is around 95 percent. The cross-validation is sharply increased with the accuracy of 92% to 97% when training set size was 200. The accuracy is continuously increased when the training data is increased. The ROC curve demonstrates good results where the AUC is 0.8, which is a good separation of the flooding and legitimate classes.



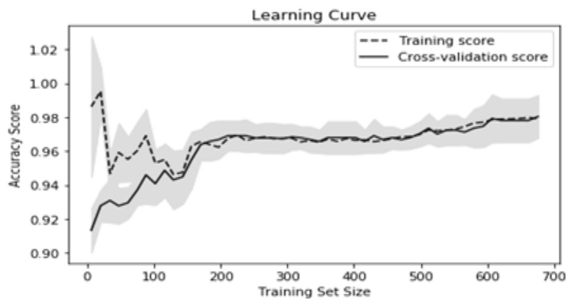


Figure 5-10 AUC value of 0.8 is a good separation of classes, as well as Learning Curve, converges on lower training set instances.

Multi-Layer Perception is a FNN, which generates a set of outputs characterized by multiple layers of input that are connected as directed graphs between input and output layers [226]. The evaluation metrics for the testing set is greater than 97% which demonstrates the effectiveness of the training model. The results of the evaluation metrics are given in Table 5-9. Here the cross-validation score follows a zig-zag pattern with sharp downward peaks at two points as given in Figure 5-11. The accuracy is thoroughly increased with the increase in the training set size from 91% to 95%. However, when the training set size reached 400, the accuracy suddenly dropped. This pattern is observed due to the variations in the accuracy of the training set which subsequently affected the cross-validation. The ROC curve represents excellent results where AUC is 0.92, which corresponds to the perfect separation of the flooding and legitimate classes.

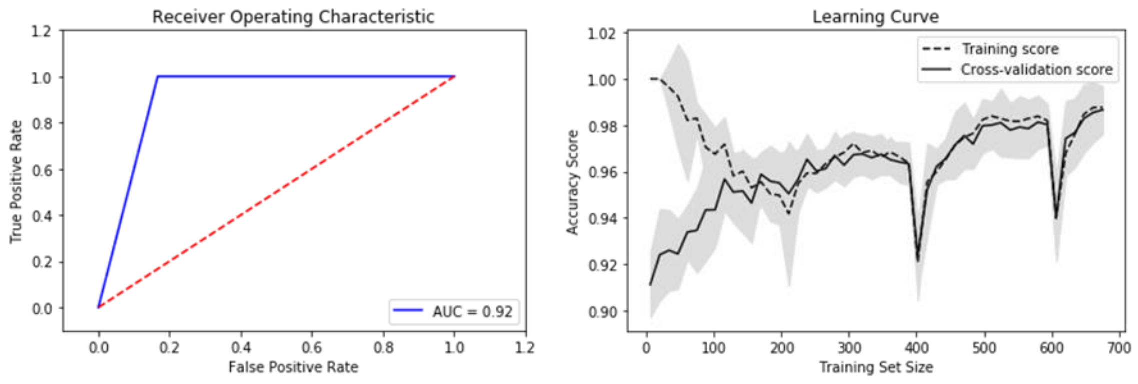


Figure 5-11 A perfect ROC and variations in the accuracy of the trained model in Learning Curve.

### 5.4.4 *Naïve Bayes and Random Forest*

Naïve Bayes uses Bayes algorithm to classify the objects which uses the concept of naïve or strong independence among the attributes of data points [227]. The evaluation metrics of the trained model can be observed in Table 5-9. The accuracy score is 92% for the test set, which is lower than the previous discussed algorithms. In Figure 5-12. the Learning Curve converge at around 500 training instances which demonstrates that an increase in training instances will have a positive effect on the performance of the classifier. The ROC curve demonstrates excellent results where AUC is 0.92, which is a perfect separation of the flooding and legitimate classes.

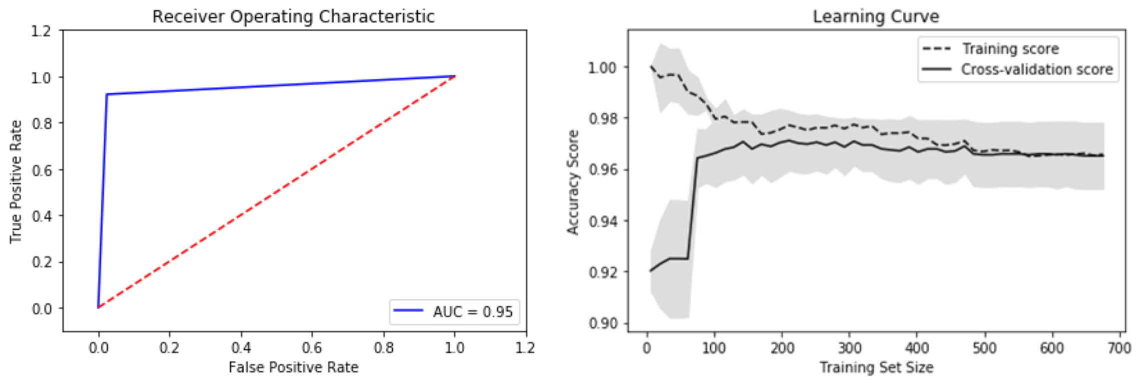


Figure 5-12 An excellent score of 0.95 of AUC, in the same way, increase in the training instances have a positive effect on the validation of the classifier.

Random Forest algorithm develops random decision trees by analyzing a set of variables and providing the output of the class that is the mean prediction of the individual decision trees [228]. Table 5-9 illustrates the evaluation results of the Random Forest model on the training and testing set. In the Random Forest classifier, the evaluation metrics for training and testing were 1.0 which illustrates the excellent efficiency of the trained model in accurately classifying the dataset. The ROC curve describes exceptional results where AUC is 1, which is a perfect separation of the flooding and legitimate classes. The accuracy of the cross-validation curve increases continuously and converges with the training score at around 600 instances of the training set.  The results are given in Figure 5-13.

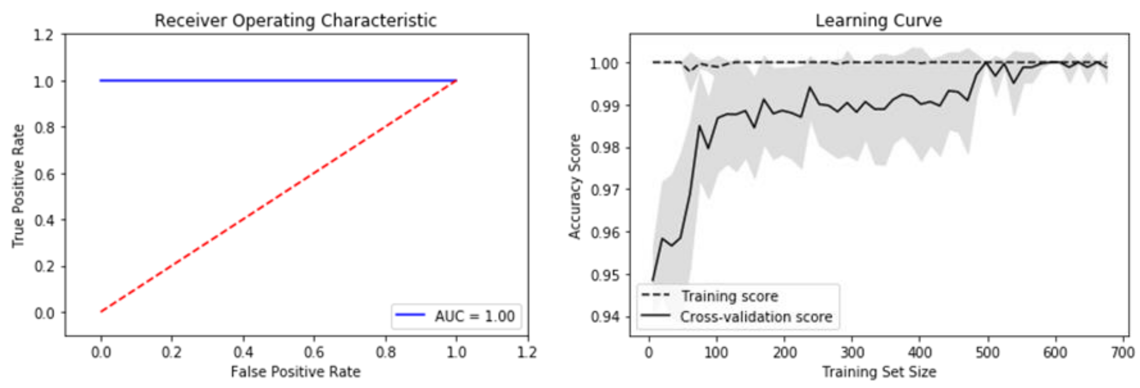


Figure 5-13 A perfect AUC score on ROC while in Learning Curve, training instances are playing a positive role in the accuracy of the classifier.

### 5.4.5 *Stochastic Gradient Descent and Support Vector Culstering*

This is an iterative algorithm to optimize a differential objective function which is a stochastic approximation of gradient decent optimization [229]. In Stochastic Gradient Descent classifier, the evaluation metrics were 1.0 which is a perfect result for accurately classifying the dataset. The ROC curve demonstrates excellent results where the AUC score is 1. The Learning curve illustrates that initially, the accuracy of the test set was lower and with the increase of training data the accuracy of the test set was increased and reached 100% when the cross-validation converged at 400 training instances. The evaluation results of training and testing are given in Figure 5-14.

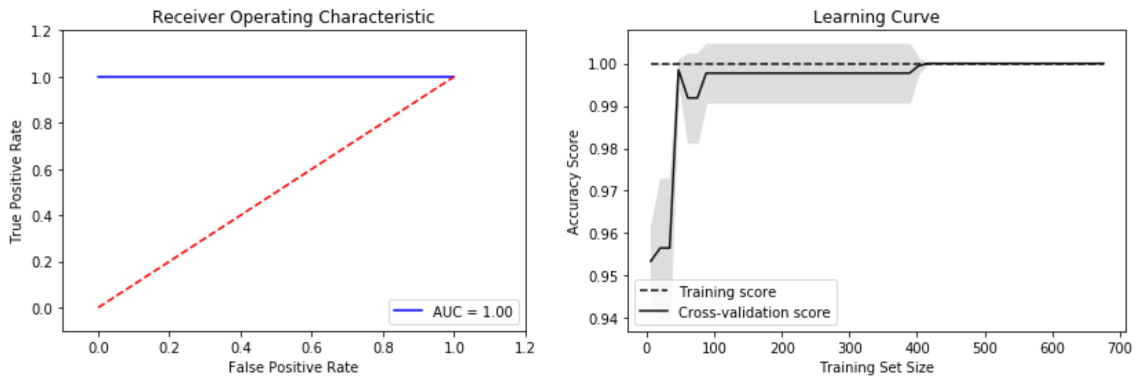


Figure 5-14 The evaluation of SGD.

Support Vector Clustering is a classification algorithm which sorts the data into one or two categories and constructs a map of the sorted data by adding a margin between them as far as possible and classifies an unknown data to which data point it will belong to [230]. Table 5-9 presents the evaluation metrics of SVC algorithm; the Accuracy score is 0.89 which is the lowest accuracy in all the implemented models. Overall the evaluation results of this classifier were poor. The region under the ROC curve is 0.62 given in Figure 5-15, illustrates the poor separation of flooding and legitimate class. The Learning Curve converges at around 300 training set instances.

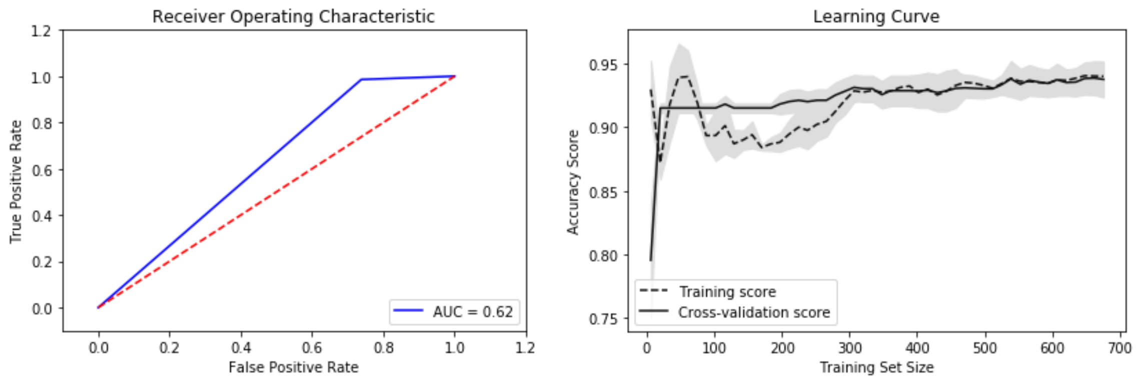


Figure 5-15 The AUC value of 0.62 illustrates the power performance in distinguishing between flooding and legitimate classes.

We compare all the algorithms after training and testing and results are given in the Table 5-9. It can be observed that Adaptive Boosting, Decision Tree, Random Forest, and Stochastic Gradient Descent performed the best among all the algorithms. Subsequently, we can also observe that the KNN produced results close to the value of 1. There is a vast difference among performance metrics of Naïve Bayes, as the values of F1Score is 0.9288 as compared to Accuracy which is around 0.9803. MLP also performed better, however, the accuracy values of SVC were the least among all the algorithms.

## 5.5 ML Solution for CyberPulse Traffic Classifier

In the section above, it is clear that different algorithms responded differently, and we were able to get good results from Adaptive Boost, Decision Tree and SGD. Upon further research ANN turned out to be a strong candidate for CyberPulse solution. It is a widely used technique for classification tasks it deals with the connected group of nodes called neurons. The input features are represented in terms of nodes that are mapped to the neurons in the ANN layers. It contains multiple layers where the outputs of different layers are passed to the next layer which performs some computations moreover, an activation function is used. The non-linearity is applied on the data usually at the output layer. The backward propagation and SGD algorithms are used to train the ANN which transforms the computation into the output. The ANN accompanies many advantages where some of them are given below.

- Due to the use of numerous intermediate layers, ANN can model complex non-linear relationships in an effective manner. Moreover, it provides precise results which makes them an optimal choice for the flood traffic classification problems.
- The structure of the data does not matter highly in ANNs, in this way, they can be used with any type of relationships involving the variables.

Keeping in view this discussion, ANNs seemed suitable for use in the CyberPulse LFA traffic classification solution. However, following are some important points to note:

- Because of the complexity of these models, they're not easy to interpret and understand.
- ANN can be quite challenging and computationally intensive to train, requiring careful hyper-parameter tuning and setting of the learning rate schedule.
- ANN requires a lot of data to achieve high performance and are generally outperformed by other ML algorithms in "small data" cases.

Looking at the limitations, we reached to a conclusion that CyberPulse ML classifier should be dataset and algorithm agnostic, supporting any pre-trained number of models up to 10.

The design should allow passing the input traffic through these models simultaneous, hence a multi-threaded design pattern should be adopted.

## 5.6 Chapter Summary

LFAs employ low-rate legitimate looking traffic and hence pose a unique challenge for identification. Through experimental study in this chapter, we were able to ascertain that ML is an effective technique in the detection and mitigation of LFAs in SDNs. Using the detailed analysis presented, we were able to select a suitable dataset for ML model training. We also reached to a conclusion that CyberPulse traffic classifier component needs to be a multi-threaded engine that supports up to ten pre-trained models; making it dataset and algorithm agnostic. Additionally, the research questions 7 and 8 were adequately answered. In order to help better understand the CyberPulse evaluation results presented in the upcoming chapter, a brief preliminary study of ML algorithms was also presented in this chapter along with pointers on the selection of a right algorithm depending upon the problem scenario. This section of the thesis helps build the CyberPulse ML classifier and provides necessary details along the way such as dataset and features explanation, preprocessing and the steps involved in training models. These details ensure that this research remain re-producible.

CyberPulse user will essentially need to train models and therefore a comparative study of tools along with coverage has been discussed. Pytorch was found to be fit for purpose in this context. The 10 trained models were tested without first fitting in the developed prototype, that helped tune and optimize certain characteristics for best results. These ROC and learning curve graphs and the data visualizations are not presented in the CyberPulse evaluation chapter, as we believe these are part of the CyberPulse development phase and are best suited here in chapter 5.

The outcome from this chapter made the basis for us to believe that creating a multi-threaded CyberPulse prototype will indeed be a good idea that could work on ten trained models in parallel and yet work at the line speed. Overall this chapter provides a comprehensive discussion on selecting optimal ML algorithm and ML tool for training models and data visualization. These trained models are provided as an input to CyberPulse to help it classify traffic at real-time.

In the next chapter, we aim to connect all dots and discuss the CyberPulse experimental setup and implementation. We will comprehensively demonstrate the evaluation of the CyberPulse using carefully designed performance metrics.

# CHAPTER 6. EXPERIMENTAL SETUP AND EVALUATION

SDN facilitates centralized management of the network where the network policies, security, and management can be implemented in a granular way. The network can be programmed to respond to certain types of events. SDN has a holistic view of the network which can help to measure the traffic features at any time. These statistics can be utilized to identify the malicious and adversarial flows. Traditional networks need hardware to perform sophisticated network measurements. However, SDN provides flexibility to perform granular network traffic measurements which can be utilized by the network programmers to develop customized applications. SDN provides an opportunity for application developers to develop applications to manage SDN in a customized way.

It provides application developers more visibility into the organization's network. The controller provides a single security point which can provide insights into the impending threats. The programmability in the network facilitates the application developers to develop customized applications and manage diverse devices without knowing the complex internal working mechanism of the network hardware. By finding the locations where the congestion or security vulnerabilities are occurring, network outages can be prevented before they occur. Experimentation is necessary for the evaluation of any developed solution. The experimentation needs a realistic setup that will be deployed in the actual network paradigm. Therefore, we setup an experimentation testbed to evaluate CyberPulse using mininet and Floodlight controller.

We explain how useful CyberPulse is in dealing with the LFAs in the SDN scenario. In this chapter, we comprehensively discuss the experimental setup of the current research including the network topology, parameters for topology creation, and operational workflow of CyberPulse. We will discuss CyberPulse deployment and parameters for traffic generation and attack hosts selection. We provide an in-depth explanation of how to download, setup, and deploy CyberPulse framework for the users how don't have any prior experience in SDN security. Subsequently, we provide detailed evaluation study using different experimental scenarios. We initially evaluate CyberPulse using ANN algorithm and provide detailed comparisons of our results with four different algorithms. Finally, we evaluate CyberPulse using different performance parameters including detection time, link degradation, number of attackers, and throughput. We also evaluate CyberPulse using different algorithms and provide the results.

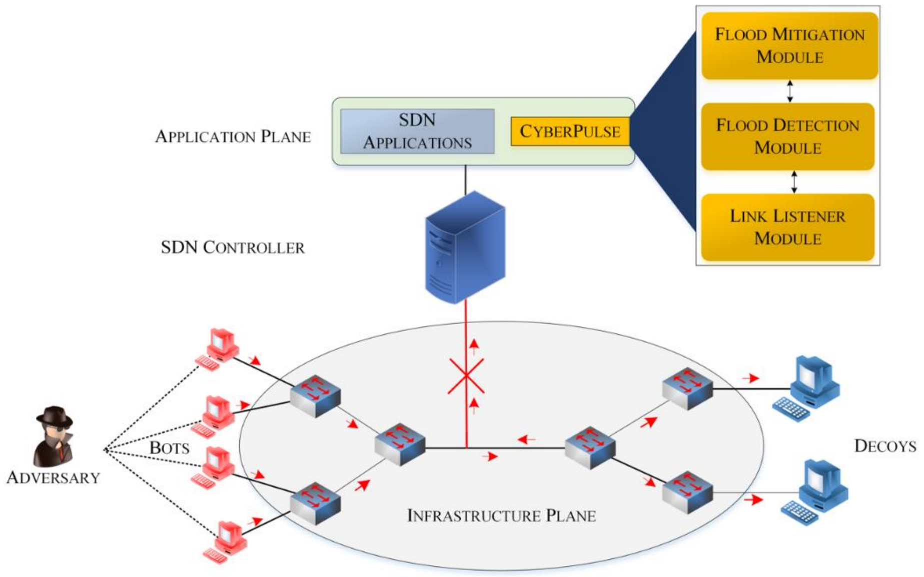


Figure 6-1 LFA setup and deployment in the application plane of SDN.

## 6.1 Prototype Parameters

Figure 6-1 describes the implementation of CyberPulse at the application layer of the SDN controller. We integrate all the modules developed in python to perform the experimental evaluation. In order to generate traffic for experiments, we deploy mininet. Initially, we will provide the configuration of the testbed and subsequently, a detailed explanation of the development and deployment will be discussed. This chapter also includes the tools used and the workflow of CyberPulse. This prototype works as an independent system in the SDN ecosystem to enable network surveillance against LFAs. It provides a comprehensive paradigm in the network security it has the following advantages.

- Network status information
- Automatic detection
- Flexibility to use any algorithm
- Flexible to use any confidence interval
- Selection of the links and port numbers
- Any remote controller selection
- Customized program duration selection
- Customized link capacity specification
- Selection of traffic generating hosts and servers

With the above-mentioned characteristics, CyberPulse becomes a sustainable choice against LFAs. It can provide a holistic solution against LFAs which was not previously present to the best of our knowledge. Early LFA detection is necessary in devising a solution against these attacks. CyberPulse devises a solution that incorporates pre-trained algorithms; hence, the overhead of the real-time training can be minimized. Moreover, the mitigation solutions need to provide 24/7 services. Hence, CyberPulse can work with SDN all the times and provide surveillance.

Every solution against LFA must not affect the legitimate traffic. Alternatively speaking, the false positive rate must be low. CyberPulse has been tested with many network configurations, traffic parameters, and mitigation algorithms. It puts a very less performance load on the infrastructure and has a very minimal overhead on the network. The onboarding of a mitigation solution involves the alignment of the solution with the underlying process model. It ensures that all the components of the solution are up and in a synchronous state while prior to performing the mitigation thereby does not take much time for the setup to initialize. CyberPulse in this regard, perfectly aligned with the SDN ecosystem and can provide an effective defense against LFAs in SDN.

### *6.1.1 The Topology*

We implemented a prototype of our system using five servers each connected to an OF Switch. We developed the solution in python which uses mininet for traffic generation. Initially, the attack is launched and the iperf server send LFA traffic to the destined server. Network topology was designed as presented in Figure 6-2. It is a linear topology which includes 5 hosts and 5 OF switches. The remote controller running on a Windows machine is utilized and accessed from the Ubuntu machine using port 6634. The host 1 was used to send low-rate legitimate traffic on the network using iperf tool. The emulated network topology consisted of the parameters given in Table 6-1, the values of jitter and delay were 0.

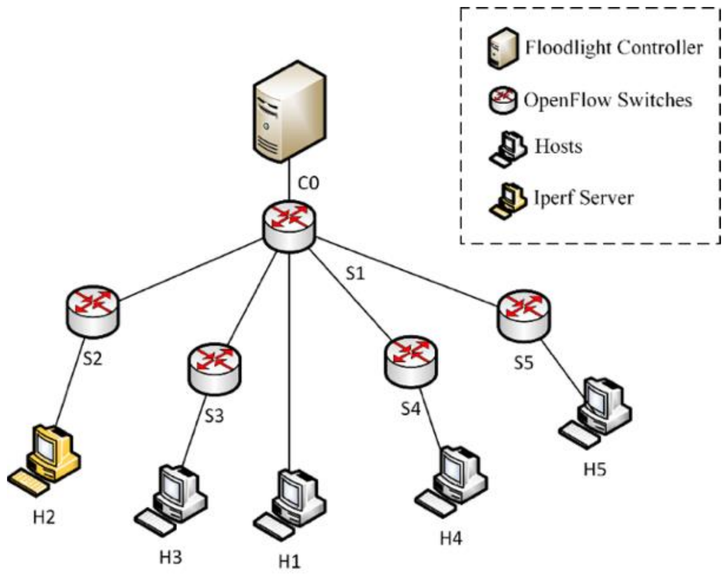


Figure 6-2 Network topology of CyberPulse.

CyberPulse was developed as a security solution on the application layer of SDN controller. The process starts by running floodlight controller [231] on the Ubuntu machine. By default, the statistics collector is turned off in the floodlight controller so, we need to turn it on. We can also adjust the statistics collection interval timing. CyberPulse is developed in python and it can be started from a separate terminal. A separate JSON configuration file is incorporated so that the parameters of the prototype can be changed according to the desired set up.

Table 6-1 Parameters for topology creation for evaluation.

| **Parameter** | **Value** |
|---|---|
| Switch Count | 5 |
| Hosts per switch | 1 |
| Controller Port No | 6653 |
| Link Capacity | 1.25 Byte |
| Mininet OS | Ubuntu |
| Controller OS | Windows |
| Flooding threshold | 10000 |
| Link Bandwidth | 10 Mbps |

### *6.1.2 ML Configuration*

In the CyberPulse security solution, the ML parameters can be adjusted by a JSON configuration file. Table 6-2 contains the parameters that can be supplied for ML configuration. The model name comprised up of the algorithm to use out of 10 available algorithms given in Figure 6-3 which is supplied in a *.pkl* file. Confidence values according to the precision requirement can be supplied, however it must be noted that increasing confidence interval can result in an increase in the false-positive rate. Action parameter comprised of the decision of dropping the host. We perform the experiments on different datasets to train the selected classifiers. Based on the accuracy of accurately classifying flood traffic, we selected UCI ML repository for training [232]. UCI dataset was providing the best results for classification where the false positive rate was also low.

Table 6-2 Parameters for ML module.

| **Parameter** | **Value** |
|---|---|
| Model Name | The name of algorithm |
| Confidence level | Confidence level of detection |
| Action | Action to be taken (e.g. drop host) |

### *6.1.3 Traffic Monitoring*

A stat collector module was developed in python which loads the trained model in a *.pkl* file format. It then gets the traffic flow ID, switch list, and Node ID and then reads the flow statistics and define the flow classes according to the thresholding criteria given in the training set. Subsequently, it reads the packets and switch port statistics of the network traffic. It then collects all the statistics defined in the Table 6-2 using the switch and port statistics APIs exposed by the floodlight controller.

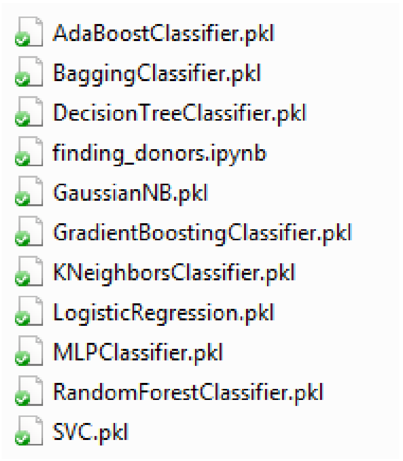

AdaBoostClassifier.pkl
BaggingClassifier.pkl
DecisionTreeClassifier.pkl
finding_donors.ipynb
GaussianNB.pkl
GradientBoostingClassifier.pkl
KNeighborsClassifier.pkl
LogisticRegression.pkl
MLPClassifier.pkl
RandomForestClassifier.pkl
SVC.pkl

Figure 6-3 A screenshot of the generated .pkl files.

In the next step, it saves all the statistics in an array and subsequently, it evaluates the shape of the array using *python NumPy.reshape* method. The extracted statistics are applied with the ML model with a desired confidence level and the graph is plotted according to the network status information.

## 6.2 CyberPulse Overview

Figure 6-4 provides an overview of CyberPulse security framework, it can be observed from the figure that Floodlight is connected with data plane devices using southbound API. One malicious host is also present which is generating flood traffic on the network. The statistics measurement module is extracting network statistics from the Floodlight controller using REST API. The classifier contains pre-trained classification algorithms which get traffic features from the statistics measurement module. It classifies the network traffic by given confidence interval and algorithm, it then transfers the results to flood defender. Finally, the flood defender module drops the malicious flows by updating rule in the OF switches. The network status monitor plots surveillance graphs by getting information from the statistics measurement and flood defender modules.

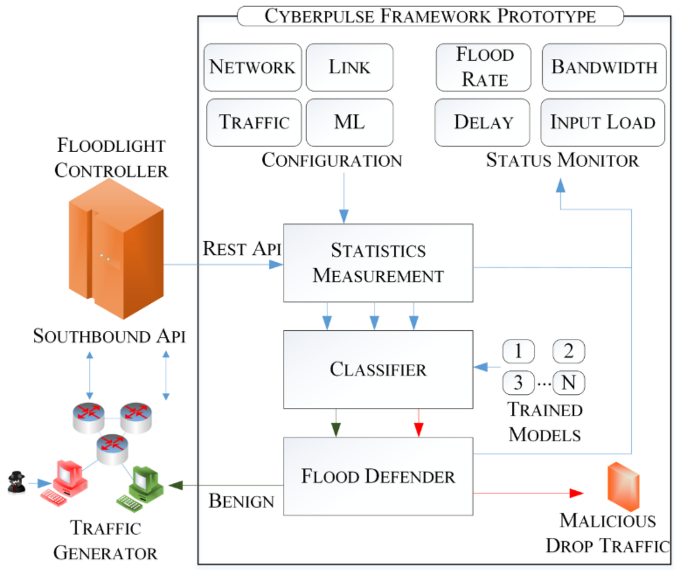


Figure 6-4 CyberPulse overview.

## 6.3 CyberPulse Working Implementation

A mininet testbed provides network researchers to simulate threats in a secure environment and to develop, simulate and deploy security solutions. This subchapter includes the procedure of CyberPulse development, we discuss the architecture, its, modules, configuration and the detailed procedure to run and setup. We have developed a security solution for SDN simulating LFA and developed a prototype. In CyberPulse we use python to generate traffic, collect statistics to train the ML model, test the traffic, and visualize the results.

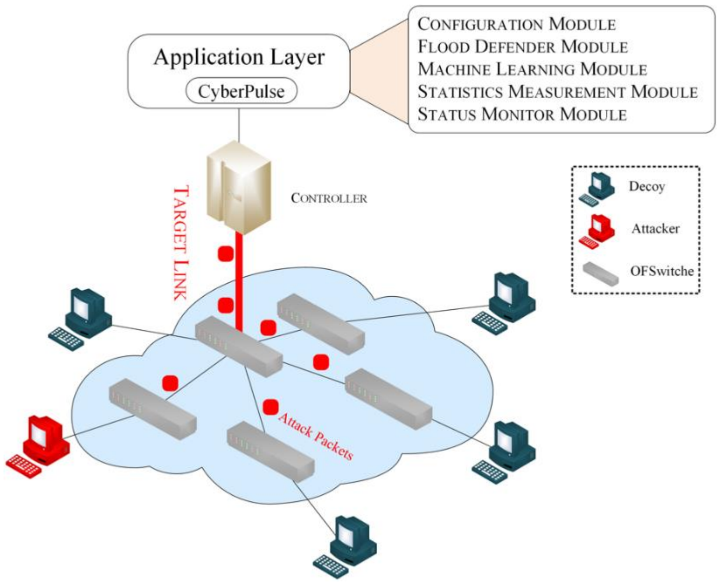


Figure 6-5 CyberPulse implementation at application plane.

Figure 6-5 provides the implementation of CyberPulse at the application plane of SDN. CyberPulse is developed in two phases: the ML training system which provides interface to train ML classifiers using the training dataset and outputs the *.pkl* files. The *.pkl* file is an output file generated by python which enables files to be serialized onto the disc and de serialized back using byte stream which represents the objects. The *.pkl* file is generated to save network overhead of transferring huge files. Python *dumb ()* and *load ()* methods were used to create and load *.pkl* files respectively. The ML training system train 10 algorithms discussed in the previous chapter. The respective algorithm can be loaded using *load ()* method of python. Figure 6-6 contains the flow of operation that CyberPulse uses to perform the network surveillance operation.

CyberPulse starts by providing network configuration including topology, traffic generation, and ML parameters selection. When the network is configured and deployed, the Link Listener starts to collect statistics of the flow. The Link Listener collects the statistics keeping in view all the available resources including, flow, switch, and packet statistics. Link Listener forwards these collected network statistics to the classifier module which consists of pre-trained algorithms selection containing the information of the source *.pkl* files. It is to be noted that only one algorithm is used for the classification of network traffic.

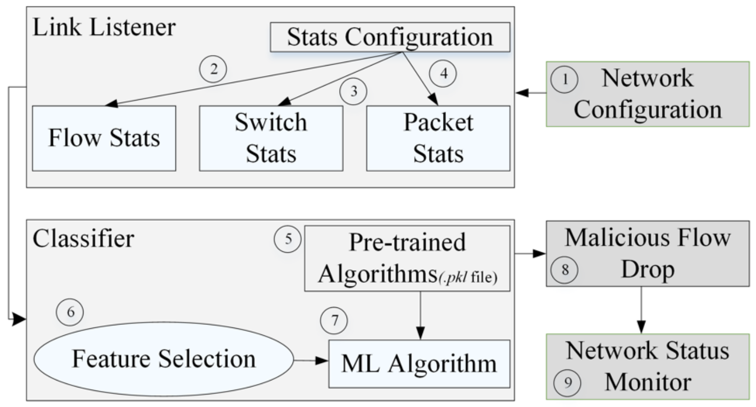


Figure 6-6 The flow diagram of CyberPulse.

The classifier module identifies the malicious flows using the selected classifier and confidence rate. The malicious flow drop module performs the classified flows elimination. In the same way, the network status monitor draws real-time network graphs. This module concurrently runs with the start of the network and draws the real-time network information on the screen. This module efficiently demonstrates the impact of LFA on different parameters of the network including, bandwidth, file transfer, packet drop rate and the effect of a number of attackers.

The traffic generation module receives an input from a user via a JSON configuration file containing the topology information, iperf clients, and ML configuration. Detailed information about JSON configuration is given in Figure 6-9.

In this file the input information is categorized in different sections, the first category contains the information regarding, controller management, it assigns the controller IP, controller port and REST API port numbers. In the next category, statistics collector information is collected in which we can input program duration, collection interval, and flooding threshold values. ML configuration category of the JSON file contains the model name and the confidence level at which the ML module will perform the LFA mitigation. The confidence interval can be given from 0-100 depending upon the required accuracy of LFA mitigation.  In the traffic configuration, link capacity, switch count, host per switch, iperf server, and traffic duration can be specified. Furthermore, specifications of *iperf* server can also be adjusted here.

## 6.4 CyberPulse Deployment

CyberPulse can be deployed on Ubuntu machine, by installing following pre-requisites. The source files of CyberPulse can be downloaded from the link given in the appendix of this report. The prerequisites for CyberPulse deployment are given in Table 6-3.

Table 6-3 The prerequisites for running CyberPulse.

| **Object** | **Version** |
|---|---|
| Ubuntu | 16.0.4 |
| Java | Runtime 8 |
| Floodlight Controller | 1.2 |
| Anaconda | Any version |
| Mininet | latest |

The floodlight controller is run first on a separate terminal and on the other terminal the CyberPulse source files are loaded. CyberPulse is started using */run.sh* command which starts initially by creating the network topology, assigning the link bandwidth, jitter and delay. Then it checks the connectivity of the components using ping command. Subsequently, the traffic is generated by the *Iperf* server and flooding of the links is started, in the meanwhile statistics collector start gathering statistics of the network and classification is started. The flood drop module drops the flows identified by the classification module. Finally, the network status monitor draws graphs for the visual analysis of traffic on the network.

For statistics collection using Floodlight controller following parameters need to be supplied in the floodlight resource files as represented in Figure 6-7. Code in Figure 6-8 needs to be run in sequence to start the operation of CyberPulse. The configuration of the JSON configuration file has been given in Figure 6-9.

```
file src/main/resources/floodlightdefault.properties set
net.floodlightcontroller.statistics.StatisticsCollector.enable=FALSE
net.floodlightcontroller.statistics.StatisticsCollector.collectionIntervalPortStatsSeconds=10
```

Figure 6-7 Floodlight statistics collection setting.

```
cd floodlight-1.2; make
Initialize:
Sudo rm –rf /var/lib/floodlight/
Run:
Cd floodlight-1.2;
Sudo java –jar target/floodlight.jar
Run project
Unzip mlsdn.zip
Cd mlsdn
./run.sh
Check status |
Tail –f Statcollector.log
```

Figure 6-8 The sequence of commands to operate the prototype.

## 6.5 Experimental Setup

In this section, we perform extensive simulation and analysis to evaluate the performance of CyberPulse in two scenarios, i.e. in a flooding attack scenario and in normal traffic scenario. Traditional SDN environment, offers no flooding attack defense mechanism, we will demonstrate how CyberPulse is an effective extension in SDN environment.
A virtual network was designed using a single desktop computer to implement CyberPulse in an emulated network environment. We run our experiments on a PC with Intel(R) Xeon(R) CPU E3-1225 v5 @ 3.30GHz and 16 GB RAM. Windows was running on the host machine and Ubuntu was running on Oracle VM VirtualBox. We used floodlight OF controller which is well known OF controller written in java programming language.

### *6.5.1 Network Model*

We created network topology as presented in Figure 6-10 as it can be observed from the Figure that it is a tree topology with three levels consisting of 8 hosts and 7 OF switches. We interfaced the remote controller running on the windows machine with mininet using port 6634. We used iperf tool to send network traffic to hosts. We plot graphs to illustrate the impact of flood traffic on the links. In the next step, we train the ML model using flood attack dataset provided by UCI ML repository the emulated network topology consisted of the following parameters.

- Eight virtual hosts each having unique IP address
- Seven OF software switches
- OF switches were configured to connected with a remote controller
- The bandwidth of all the link was set to 10 Mbps

- Floodlight remote controller, running on Windows machine

```
{
   "controller":
    {
        "controller_ip": "127.0.0.1",
        "controller_openflow_port": "6653",
        "controller_rest_port": "8080"
    },
    "stat_collector_config":
    {
        "programme_duration_min":1,
        "collection_interval_sec":5,
        "flooding_threshold_byteCount":1000
    },
    "machine_learning_config":
    {

"model_name":"ML/Pickle_files/LogisticRegression.pkl",
        "action":"drop_host",
        "confidence_level":80
    },
    "important_link":
    {
        "switch_id":"00:00:00:00:00:00:00:01",
        "port_number":1
    },
    "traffic_config":
    {
        "link_capacity_in_MByte":1.25,
        "switch_count":5,
        "host_per_switch":1,
        "iperf_server":"10.0.0.1",
        "traffic_duration_min":1,
        "iperf_client":[
            {"host_ip":"10.0.0.2"},
            {"host_ip":"10.0.0.3"},
            {"host_ip":"10.0.0.4"}
        ]
    }
}
```

Figure 6-9 JSON configuration file.

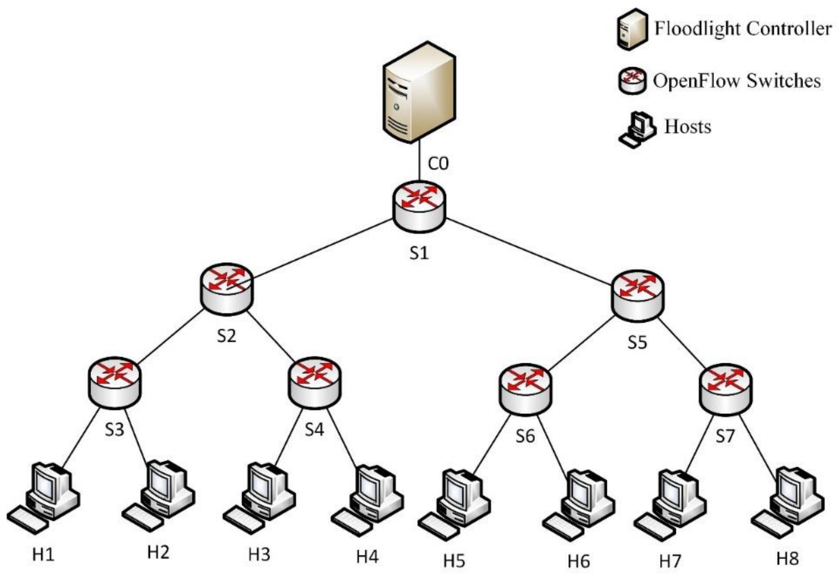


Figure 6-10 Emulated topology using mininet and Floodlight controller.

### 6.5.2 *Flood Traffic Manipulation*

The test-bed network comprised up of 7 OF switches, and 8 hosts, connected with the controller each having link capacity of 100 Mbps. We used iperf tool to send legitimate traffic to different hosts. Iperf is an open-source tool to send and measure network performance by sending TCP traffic.

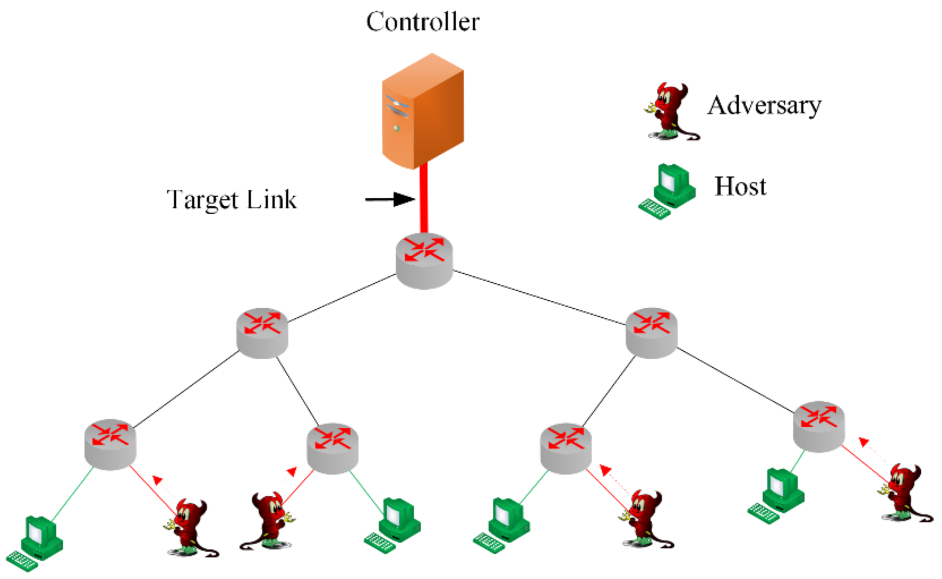


Figure 6-11 Adversary model to attack the control channel.

We set one host as a server and one as a client and sent TCP traffic over the network. There are various parameters in iperf that can be changed for measuring different network metrics. We used Hping to generate flood traffic and analyze the impact of the traffic on the links. Figure 6-11 illustrates how we utilized some hosts as adversaries in our experiment. We perform experiments by sending flood traffic towards target servers and analyze link bandwidth after sending this traffic. It can also be noted that network bandwidth was set to 1000 Mbps at the time of topology creation in mininet. Following command was used to create a tree topology and fixing the bandwidth of the links to 1000 Mbps.

```
sudo mn --controller=remote, ip <ip of windows machine> --topo=tree,
3 -link tc, bw=1000
```

In our experiment, we used some hosts as attackers as presented in Figure 6-11, which were sending *Hping* traffic towards targets links. We send attack traffic by opening a separate terminal for every attack node. We used *iperf –s* command to start a TCP connection server. Using this command, a host will start listening on *TCP port 5001* subsequently any host that need to connect to server can connect to this server using *iperf* `–s –c 10.0.0.1 –t 15` command. This command starts a host to connect to the server with an IP address of *10.0.0.1* and time limit of *15* seconds. At the end of the connection a summary of data transferred and bandwidth information 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.

### ***6.5.3 Attack Simulation***

We assumed hosts in the network as decoy servers which are manipulated by an adversary. We set the bandwidth of all the links to 1000 Mbps and measure the bandwidth after sending and receiving attack traffic between decoy servers. We used Wireshark [233] a network packet analyzer tool, for packet capturing and analysis. We started an attack by manipulating decoy servers around a target link. The decoy servers started to send traffic. The role of decoy server, H2 was set as server and H3, H4 and H5 was set as client decoys. As it was discussed in Chapter 3 that in LFA decoy servers send traffic to each other in order to achieve target link flooding. We run the experiment by varying the number of decoys and measure the bandwidth consumption of the links. We perform the experiment with one, two, and three attackers and measure the values of bandwidth consumption.

We run the experiment multiple times and each time the reported bandwidth was extracted using Wireshark. With the increase in the numbers of the hosts, the bandwidth tends to decrease. Figure 6-12a and b illustrate the effect on link bandwidth with the increase of the number of attackers. Figure 6-12a demonstrates the impact of the increase in the number of attackers and the bandwidth saturation. It can be observed that with the increase of the number of attackers, the bandwidth started to saturate and when the number of attackers reached to 35 the remaining bandwidth reached nearly 50 Mbps, which describes the effect of LFA on the SDN control channel. Figure 6-12b illustrates the effect of the number of attackers and link bandwidth, it can be noted that with the increase in the number of attackers the bandwidth consumption also tends to increase.

Another interesting phenomenon was noted that with the increase in the number of attackers and increase of the traffic on the network, the packet loss rate also tends to increase. Figure 6-13 demonstrates the effect of the increase of attackers on the packet drop rate.

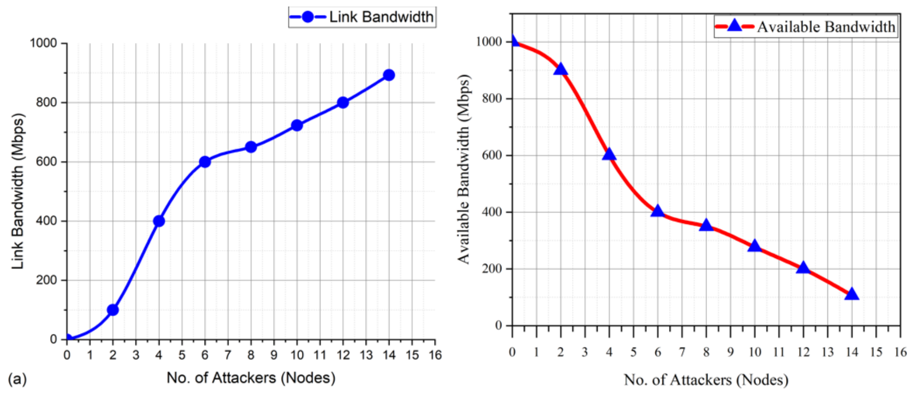


Figure 6-12 Consumed bandwidth of the network using different number of attackers.

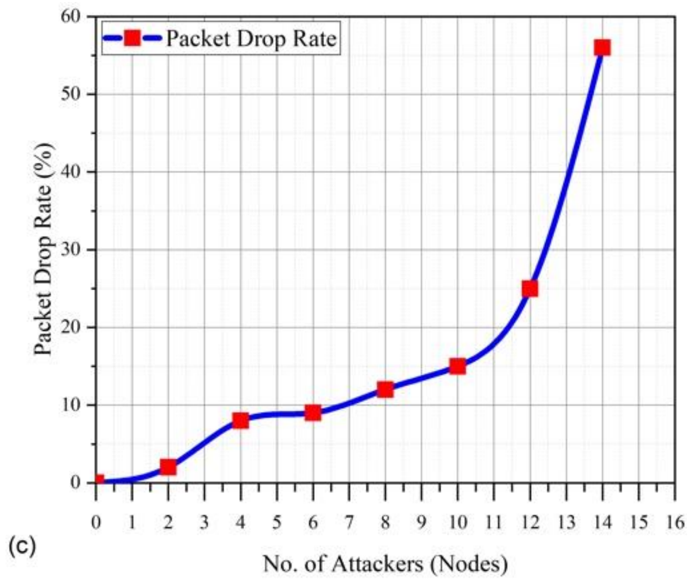


Figure 6-13 Effect of packet drop rate on the attacking nodes.

It can be observed that when the attack traffic was increased on the network packets drop rate also increased. There was a rapid vertical shift in the packet drop rate when the number of attackers increased to 7, which illustrates that when the number of attackers increased to a certain level, the packet drops rate increases. This have severe consequence on the performance of the network. And with the attackers increased to 7, there was a vertical shift in the packet drop rate. After running the experiment, we extracted flows for every node in the network. We used the training set for building the model and subsequently trained model was used to classify the statistics into benign and malicious traffic.

## 6.6 Evaluation Results

In this section, we discuss and analyze the results of our experiments. Our results are based on two scenarios first in the normal SDN traffic flow and secondly, with attack traffic. We used deep learning-based ANN technique to classify network traffic. We use UCI ML repository flooding attack dataset UCI_ML_BOS for training [184] and Impact flood attack classification datasets. The traffic was classified into two categories depending upon the severity of the attack i.e. the legitimate traffic flows and flooding flows. According to previous studies [185], [186] when the utilization ratio of the network resources, is increased more than 40 %, it indicates that the network performance has started to degrade. In a similar way, the increase in link utilization ratio also indicates that the network is under

LFA. We perform extensive experiments and evaluate our technique on three ML evaluation metrics, precision, recall, and F$_1$score.

### 6.6.1 Evaluation Parameters

We use three metrics for accuracy evaluation, 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 (6-1):

$$Percision = \frac{T_p}{T_p+F_p} \quad \text{(6-1)}$$

The results of the accuracy evaluation metrics are given in Figure 6-14. Precision values, closed to one are considered to be more accurate. It is pertinent to say that CyberPulse was able to classify the traffic correctly into three categories. The accuracy of the three categories is more than 85%. Recall can be defined as the relevant flow classification divided by the total number of relevant retrieved flows. The formula of recall is given in equation (6-2).

$$Recall = \frac{T_p}{T_p+F_n} \quad \text{(6-2)}$$

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 (6-3).

$$F_1\text{Score} = 2 \times \frac{Percision.Recall}{Precision+Recall} \quad \text{(6-3)}$$

It can be observed from precision and recall in Figure 6-14 the F$_1$Score metrics performed 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 these metrics for legitimate traffic detection was approximately the same as the values of precision and recall metrics. Our results and analysis illustrate that CyberPulse was able to accurately classify the traffic. Based on the classification values the Flood Mitigation Module was able to eliminate the flooding traffic. Overall it can be concluded that if the link flooding attacker is powerful and able to send high volumes of flood traffic than the overall bandwidth of the system will be dropped as it can be observed from the Figure 6-

12a and b. 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.

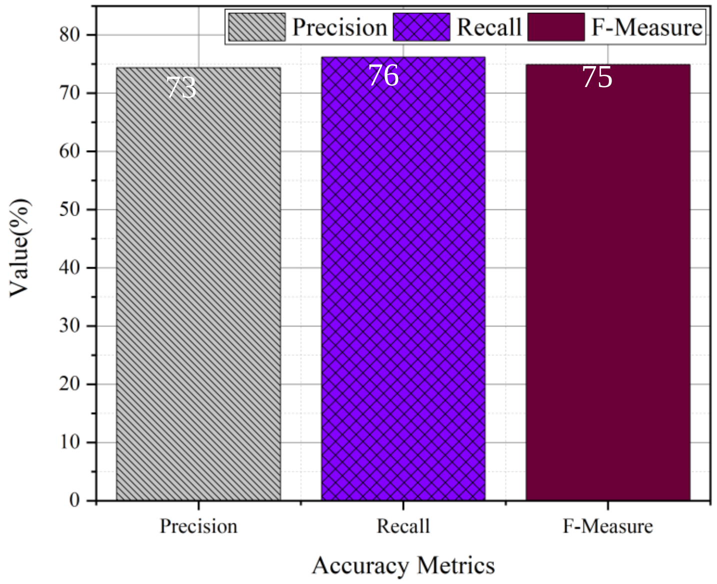


Figure 6-14 Evaluation of the ANN classifier on precision, recall, and f F1Score.

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, it classifies the flows into two categories, besides categorizing the flows into two distinct classes, it also identifies a third class, which we call it as the suspicious class.

### *6.6.2 Evaluation Using Data Partitioning*

In this section, we first evaluate the technique using different data splitting strategies. 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 an 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. As it can be observed from Figure 6-15 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 increased to over 80% which was around 74% while 50% data was used for training. This comparison provides insight for selecting a fair split of training and testing data to get better results.

### *6.6.3 Prototype Performance Evaluation Using Attribute Selection*

As it can be observed from Figure 6-15 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 these attributes.

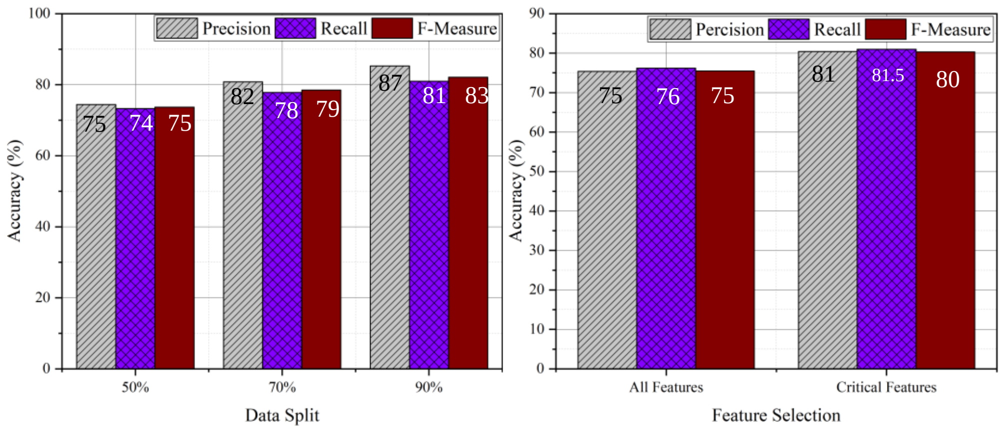


Figure 6-15 CyberPulse classifier evaluation using different data partitioning and attribute selection.

### *6.6.4 Evaluation Using Multiple Classifiers*

In this section, we provide a comparative analysis of our classification algorithm with respect to competing for 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. RF, SLR, 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 [103].

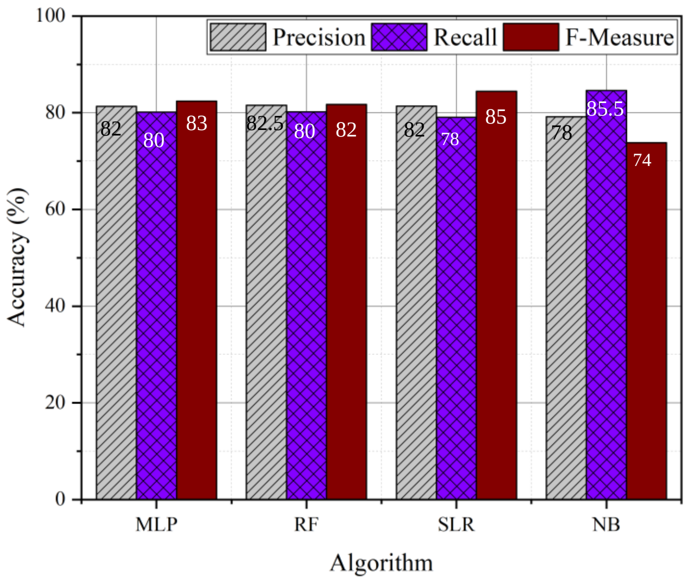


Figure 6-16 CyberPulse classifier evaluation with MLP, RF and NB algorithms.

We applied ANN classifier for flood traffic classification in our experiments. In this chapter, we compare ANN with other novel ML classification techniques.

SLR and RF have also been widely used for classification of real-time data because of their good predictive performance and excellent comprehensibility [234]. It can be observed from Figure 6-16 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 [235]. 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 requirement, MLP algorithm was the best suited as it performed well and provided reasonably good cumulative results.

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 the 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 of the network.

## 6.7 CyberPulse Evaluation Using Performance Metrics

In this section, we present and discuss the experimental results to demonstrate the performance of CyberPulse prototype in an LFA paradigm. The prototype has been developed to act as a defense against LFA on the control channel of SDN. It uses the ML-based classification methodology to identify the malicious flows and then alleviate those to safeguard the network. Multiple configuration parameters have been used to provide the required level of the defense.

To simulate a flooding attack, multiple sources were setup to send relatively low-rate traffic as compared to one or two sources sending huge files. Unfortunately, we could not find any similar study to compare our experimental evaluation. Therefore, the effectiveness of CyberPulse has been demonstrated by comparing the results in different experimental configurations. We perform the experiment with two threshold frequencies i.e. 50Kbps and 30Kbps and demonstrate the performance of CyberPulse in both the conditions and set the bandwidth of the control channel as 10Mbps. Our previous work performs the evaluation using different algorithms and illustrates that the MLP provides best results. Hence, we perform the evaluation of CyberPulse using MLP with a confidence interval of 95%. We run the experiments five times and plot the cumulative standard error during the experiments which can be observed on all the evaluation graphs. The results of the evaluation are as follows.

### *6.7.1 Evaluation Results for Delay*

The graph in Figure 6-17 demonstrates the delay incurred during LFA, the x-axis in the graph illustrates the time while y-axis corresponds to the delay incurred in Milliseconds (ms). For delay measurement, we send File Transfer Protocol (FTP) traffic over TCP which needs traffic generation tool to transfer files over the network. We consider all traffic to be part of the calculations including background and foreground traffic. When LFA occurs, the legitimate traffic faces delays which increases source to destination delivery time of the packets.

We use the ping command to measure the RTT between the hosts in two traffic intensities and measure the end-to-end network traffic delay during the attack. In the low traffic intensity, the maximum delay incurred was around 173ms. Moreover, when the attack was mitigated, the delay decreased significantly and remain around 71ms at around 90seconds time. The maximum delay observed at the increased attack frequency was around 400ms. Similarly, subsequent to the attack defense, the delay became stable around 75ms. This illustrates that the delay decreased significantly during both the experiments which illustrate the efficiency of CyberPulse against LFA.

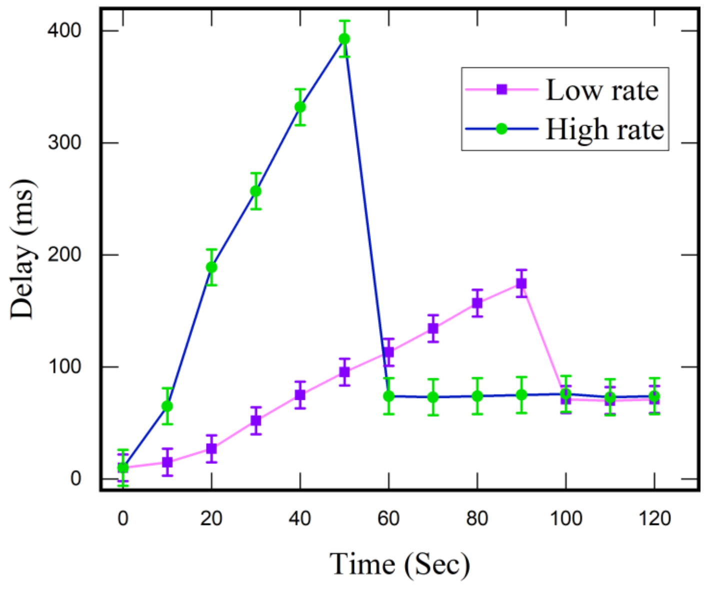


Figure 6-17 Graph illustrates how the delay is affected with the increase of flood traffic.

### *6.7.2 Evaluation Results for Bandwidth Saturation*

Bandwidth saturation measures the percentage of link capacity consumption or degradation during the attack by flooding flows. Figure 6-18 demonstrates that the bandwidth saturation increases with the increase in time during the experiment. The adversaries start to send LFA traffic towards the targets as soon as CyberPulse is executed on the Ubuntu machine terminal. During LFA, bandwidth saturation follows a proportional pattern with respect to time. When the attack was mitigated, the saturation reached to a normal level. The figure illustrates that bandwidth saturation dropped significantly when the attack was mitigated in both the attack thresholds. It remained around 4.5Mbps after the mitigation due to the legitimate traffic in the network which do not pose any threat as it was still lower than the actual link bandwidth. The impact of bandwidth saturation was continuous which can exhaust the available bandwidth of the network if the mitigation has not bee performed as it can be observed in both of the graphs in Figure 6-18

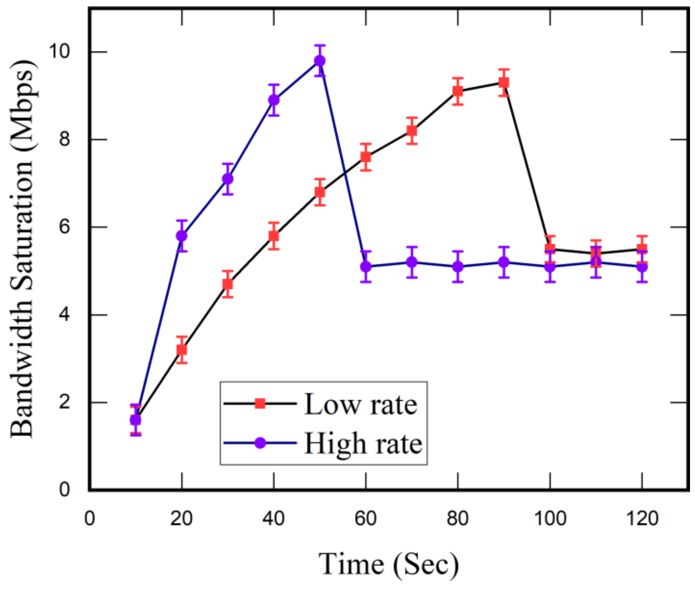


Figure 6-18 Graph demonstrates the effect of bandwidth saturation with reference to time during LFA.

### *6.7.3 Evaluation Results for Throughput*

Throughput can be defined as the data delivered per second on a specific link. In this experiment, we measure the throughput of the network while running the experiment during LFA. Figure 6-19 illustrates the effect of LFA on network throughput. It can be observed from the throughput graph that the data has continuously been delivered on the links which increased the throughput continuously. However, CyberPulse was able to detect the attack and perform the mitigation thus the throughput dropped immediately which can be observed at the peak points in the graph. throughput has been dropped at the endthe  of experiment.

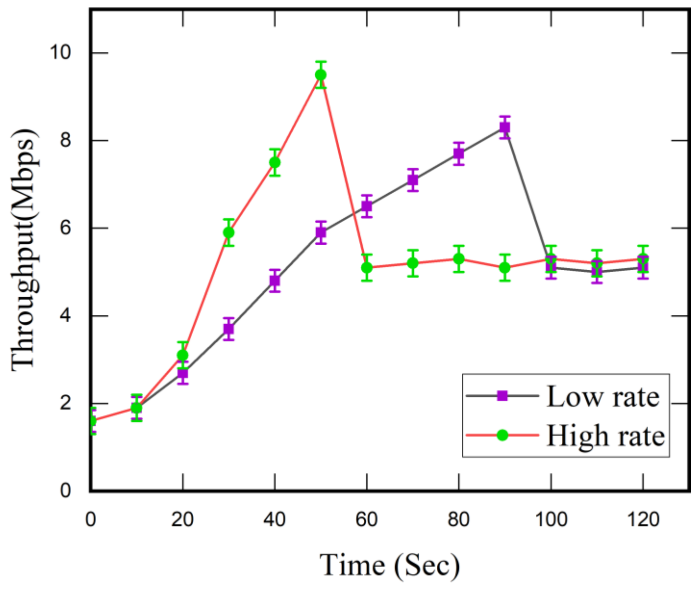


Figure 6-19 Graph describes the effect of LFA on the throughput of the network with flooding threshold of 2000.

### *6.7.4 Evaluation Results for Input Load*

Input load denotes the total traffic in the network including flooding and benign. We can observe that the input load has been increased with the increase in flooding threshold. Figure 6-20 illustrates the effect of flooding on the input load which starts to increase with the time where the flood traffic was continuously being sent on the control channel link. The reason behind the increase in the input load was that the adversary was sending carefully crafted packets which cause packet miss in the data plane switches. This packet miss instigates new flow rule installation by requesting the controller. Thus, frequent interaction with the controller causes the input load on the control channel to increase significantly. Subsequently, the attack was mitigated which can be observed at the maximum peak points in both of the graphs. Finally, we can observe that there is still a certain amount of input load on the links which is due to the benign traffic in the network. Both graphs share a similar pattern during the input load experiment.

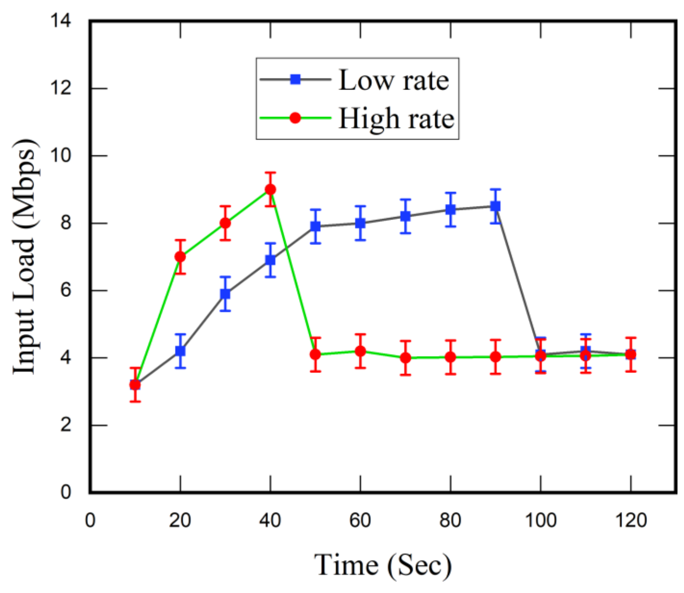


Figure 6-20 Graph illustrating the effect of LFA on input load of the network with flooding threshold of 2000.

### *6.7.5 Evaluation Results for Flooding Rate*

In this experiment, we observe the effect of LFA on the control channel of SDN. When flooding occurs, it increases the number of flow rule installation requests to the controller. Multiple such requests exhaust the control channel bandwidth and result in lowering down the performance. We run the experiment with low and high attack traffic and the results have been presented in Figure 6-21. The result of both the experiments demonstrates that the flooding rate instantaneously rises in the start and reaches at peak value. The peak has been observed because the flooding rate was increasing continuously whereas the CyberPulse takes some delay in first capturing and then classifying the flood traffic. Hence, due to the mitigation, a sudden decrease in the flooding threshold can be observed in both the graphs as presented in Figure 6-21.

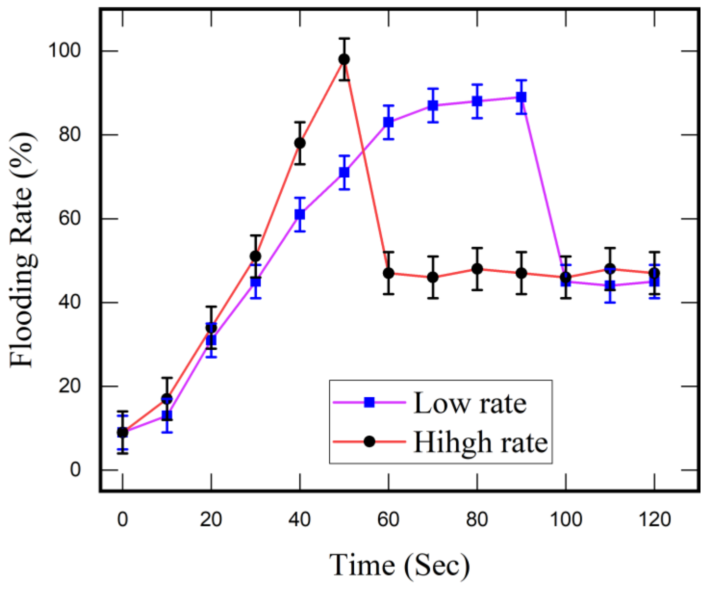


Figure 6-21 Graph demonstrating the effect of time on the flooding rate on the network.

### *6.7.6 Evaluation Results for Attack Detection Time*

When the flooding rate is low, flows in the network may not consume a large number of network resources thereby allowing them to remain undetected which increases the attack detection time. In the same way, a large number of attackers consume more network resources and can be easily detected hence, attack detection time will be low. We perform this experiment by iteratively employing attack hosts from 1 through 8, we added more traffic sending hosts in this experiment. The attack detection time with one host was around 147ms. The flooding intensity increased with the increase in the number of attack hosts due to the increase in the packet drop rate and bandwidth saturation. The traffic intensities are increased which makes the flooding flows prominent and can be detected in a shorter period of time. The attack detection time with 8 attacking hosts was around 23ms. Figure 6-22 demonstrates that with the increase in the number of attackers the detection time decreases.

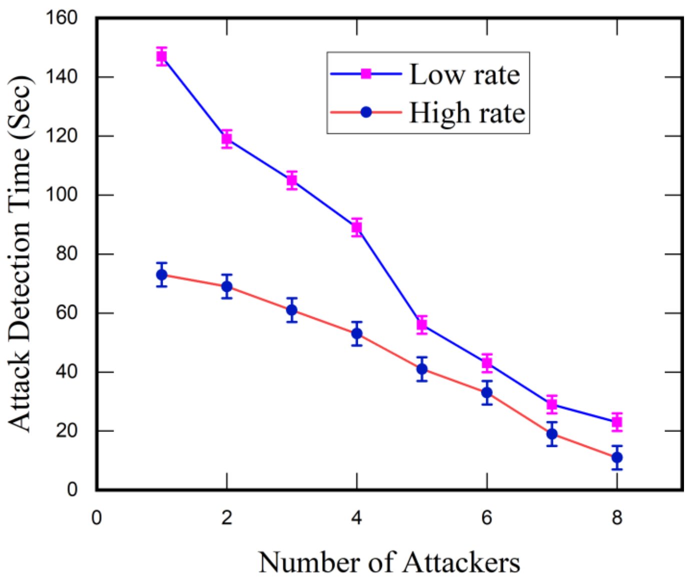


Figure 6-22 Graph describing the attack detection time with reference to number of attackers with flooding threshold of 2000.

The experimental results demonstrate that CyberPulse effectively defends LFA and provides seamless network operation. Recent solutions against LFA like LinkScope, and LFADefender [51] employing software agents to perform specialized traffic measurements which increases overhead on the network. However, CyberPulse applies SDN-based measurements to collect network statistics which poses no or very less overhead on the network.

## 6.8 Overhead of CyberPulse

Although CyberPulse employs SDN-oriented measurements, which pose a very minimal overhead. However, CyberPulse needs to compute a long range of features which increase delay in the network traffic. The experiment evaluation illustrates that CyberPulse induces 0.1 ms delay in the normal network traffic. CyberPulse has an added complexity of employing a ML approach, however, the cost-benefit trade-off is worth using this technique to safeguard current Software-Defined internet infrastructure from LFAs.

## 6.9 Chapter Summary

SDN provides an innovative mechanism in the network management for granular network programmability and control. There are a very few routers, switches, and control mechanisms available which offers the SDN-based experimentation facilities. Moreover, the existing hardware is expensive and cannot be used for a testbed. So, SDN researchers have developed a mininet tool to perform experiments and test novel SDN features they develop. Mininet allows rapid prototyping of SDN on a single computer. It is scalable for the network virtualization including hardware, processes, and control. The core characteristics of mininet are scalability, applicability, realistic experiments LFAs are hard to detect as they employ stealthy traffic to flood a target link. In this regard, we have developed a network security solution and tested using mininet.

CyberPulse is a security solution which actively mitigates control channel LFAs by first identifying malicious flows using ML-based classification technique and then dropping the malicious flows using NULL routing. CyberPulse comprises of Link Listener, Flood Detection, and Flood Mitigation Modules. In this chapter, we presented the evaluation of CyberPulse using a comprehensive evaluation parameter which proved the endurance of CyberPulse in mitigating LFA. Extensive care was taken to incorporate intended requirements of the best solution for SDN under LFA. Therefore, we developed CyberPulse considering all the required features that a state-of-the-art solution should possess.

A critical aspect of providing a solution against LFA is to get the network status at runtime. CyberPulse provides real-time network status information which enables efficient network surveillance. It provided efficient results in detection and mitigation of LFA. It has the ability to use any ML algorithm and dataset to provide defense against LFA. Moreover, CyberPulse uses an in-band control strategy where only one switch is connected with the controller, all the other switches need to use the services through the connected switch. In-band control strategy is cost-effective as providing a separate link for each switch may incur more cost and controller communication overhead. After a detailed discussion on the evaluation of the CyberPulse, we will now present the conclusion of the research in the upcoming section. The ML techniques used in this research need minimal time for classification which poses minimal overhead on the network. However, it remains close to 0.1ms which is negligible.

# CHAPTER 7. CONCLUSION AND FUTURE RESEARCH DIRECTIONS

Since the rise of cloud services, big data processing on large server farms, and changing traffic patterns, there is an increasing need for a dynamic, manageable and adaptable network architecture. The Software Defined Network (SDN) ecosystem has grown to fill this gap. After a complete revamp of the data-center network's market, SDN is now growing popular in mobile networking and wide-area networks. SDN allows seamless management of large-scale complex networks by centralizing the network intelligence into one network control component. Unfortunately, the intelligent centralization has serious underlying disadvantages when it comes to security. Link Flooding Attacks (LFAs) are considered crippling for traditional networks and are even more devastating for the SDN ecosystem.

Existing solutions lack in securing SDN from LFA which is one of the powerful forms of attack. This research presented a comparative analysis and identified crossfire link flooding technique as one of the lethal attacks that can potentially target the control channel. We presented a comprehensive analysis of LFAs on SDN and proposed a solution named as CyberPulse in securing SDN infrastructure from LFAs. Initially, we perform an in-depth study of the outcomes of LFA on all the SDN planes. Subsequently, we evaluate the effect of LFA on the popular SDN variants including SDMN, SDWN, SDLAN, and SDMNs. We perform an extensive analysis of the available literature on LFA and compare different LFA mitigation techniques. We establish that the control channel is very critical and a successful LFA has the potential to disrupt the communication of the entire network. In the absence of any viable solution in the literature, we developed a framework for securing the control channel from LFA called CyberPulse.

It consists of four modules including Link Listener Module, JSON-based Pattern Matching Module, Flood Detection Module, and Flood Mitigation Module. We explain the working principle of the modules and how they interact to accomplish the task of LFA mitigation. We explain the flow of the information in the modules and explain the design considerations needed to develop this framework. Initially, we discussed the literature on LFAs in SDNs. A Mininet testbed using Floodlight controller and JSON pattern matching technology using DFA was arranged to perform the evaluation. The JSON pattern matching engine comprised up of two major phases, compilation phase, and matching phase. The evaluation results demonstrated that despite being extendable and expressive, the performance of the proposed approach did not degrade as compared to optimally designed REs which is actually beneficial for long patterns mostly found in IDS systems. JSON approach is comparatively easier to express and extendable for new patterns due to the use of logical blocks and its unambiguous notation. CyberPulse utilized the Link Listener and JSON Pattern Matching Module to analyze the network traffic and on the basis of performed analysis, it invothe kes Flood Mitigation Module to secure SDN from LFA. CyberPulse is implemented on the application plane and it monitors, control channel to keep track of the ongoing traffic flows. For each flow in the network, CyberPulse examines the statistics on the control channel using the Link Listener Module and JSON-based Pattern Matching Module.

The pattern matching operation acts as the first line of defense by comparing the traffic packets with the underlying patterns stored in the database. It immediately drops the flow if a malicious signature is found for the compared packet in the database. Subsequently, CyberPulse has the flexibility to classify the network traffic using any ML algorithm. CyberPulse is a robust solution in a way that it operates at line speed level and alleviates the LFA during the normal network flow. The available solutions against LFA involve complex hardware for detection and defense, however, LFA is distinct in its quality that it operates on real-time traffic paradigm as well as it utilizes multiple ML classification algorithms for LFA traffic classification. To this end, CyberPulse comprises of two main components, ML training, and mininet traffic generation and defense mechanism. CyberPulse is developed after careful consideration of all the aspects of LFA.

We conducted an extensive evaluation to demonstrate the effectiveness of CyberPulse on a simulation testbed. The ML component trains 10 state of the art classification algorithms with high precision. The traffic generation and defense mechanism provide the implementation flexibility to use any of the algorithms to classify the LFA traffic and perform mitigation. CyberPulse has the capability to provide real-time network status graphs which enables efficient network surveillance. CyberPulse is a novel solution which can be deployed for highly secure network provisioning to perform the defense against LFAs. CyberPulse is highly scalable and can be deployed on the high-tech network environments. CyberPulse can be deployed in highly sensitive network installations where real-time network surveillance and defense is required.

## 7.1 Future Research Directions

For the future work, we are planning to extend CyberPulse by training the classifier on multiple datasets for multi types of security threats and perform the surveillance on a variety of attacks at the same time. This work can be extended to design a robust multi-layer and multi-types of attacks and defense mechanisms. After an in-depth analysis of research on LFA in SDN, we have identified multiple open research issues and challenges in this paradigm. Generally, a lot of work has been performed to mitigate LFA in traditional networks, however, less work has been conducted on LFA in the context of SDN. Specifically, a major research challenge is to provide LFA mitigation strategy for SDN. As the current networks are widely deploying SDN for network operation, there is a strong need to provide LFA mitigation strategy for SDN. We discuss the current issues and challenges in the following.

### *7.1.1 Control Channel LFA*

Control channel LFA is critical, so devising a solution for this channel will be a valuable scientific contribution. Following guidelines should be considered while developing a solution. Any solution for control channel LFA should be an independent application in order to pose less overhead in modifying the complex functionality of default SDN controllers. To minimize the chances of application crashing, the solution should be designed in such a way that its communication outside the network is limited to ensure its reliability.

In the current network circumstances, the traffic consistency is random, so any acceptable solution must be scalable according to the incoming traffic. In real time network scenarios, traffic inspection and analysis takes a significant amount of time so, it is difficult to provide a real-time solution for LFA. However, there is a strong need for a solution that works on a real-time basis because LFA can immediately disrupt the communication with the vital resource which can result in vital information loss.

### *7.1.2 Pattern Matching and ML against LFA*

The adversaries exhibit special traffic patterns when they attack. In this regard, the identification and mitigation of these patterns can help against LFA. Efficient pattern matching is a key technology for effective network traffic monitoring. DCI and IDS/IPS techniques also rely on identifying malicious content against predefined patterns. For providing a solution against LFA on SDN, pattern matching techniques can be designed which helps in analyzing traffic patterns on line speed. The normal traffic features can be represented as signatures and can efficiently be used to identify malicious patterns in the network. Pattern matching techniques can be utilized as the first line of defense against LFA by filtering the flood from the known attackers.

When traffic streams are processed through the pattern matching engine, the traffic can be further analyzed using ML techniques. ML techniques are widely being deployed for network traffic classification. A solution based on ML techniques can be devised to classify flood traffic on control channel. Similar to a solution provided by [168] for pattern matching in IoT, a solution can be provided for LFA attack patterns identification using historical attack data, which can be used to classify traffic flow statistics into benign and flooding flow categories. This vital identification can be further utilized to mitigate the flood traffic using any state-of-the-art mitigation strategy.

### *7.1.3 Security Against Sources of LFA*

The attack traffic in LFA is persistently sent from the adversaries because most of the times they use network of bots to send low-rate traffic. All the current solutions work by mitigating these attacks on the victim side, there is a strong need to provide solutions at the source side of LFA. However, it is very difficult to identify the sources of LFA because the intensity of traffic at the source sides is very low which makes it hard to discriminate. A better solution to alleviate the sources of LFA is through coordination between the ASs at the source and destination sides. However, the source ASs needs economic incentives to collaborate with the destination ASs. Source side security solutions can also be provided by implementing highly secure authentication services to identify and alleviate malicious hosts.

### *7.1.4 Need for Robust Solutions*

A much-needed future direction is to develop a solution to mitigate LFA in SDN as an application. Most of the current controllers use REST API to communicate with the physical hardware's in the network. REST API can be utilized to collect flow statistics which subsequently can be used for surveillance of the control channel. With the rapid developments in the field of information and communication technology, the need for reliable networks have been increased. LFA has become one of the most dangerous attacks on the networks. So, there is a strong need to provide highly robust defense against these attacks. Most of the available solutions for LFA are reactive in their implementation.

With the advent of big data technologies most of the organizations are operating online, hence, strongly secure networks are preferred by the organizations. Therefore, the need for proactive and reliable solutions have been increased. Most of the current solutions against LFA are tested on simulation methods which do not guarantee their effectiveness if deployed in real network environments. Therefore, there is a strong need to provide solutions evaluated in real testbeds. This way the requirements of scalability, complexity, and real-time accuracy can be validated.

### *7.1.5 Need for Physical Testbed Implementations*

There is a need to implement CyberPulse on a physical testbed and to extend it by training on multiple datasets for multiple types of attacks while simultaneously performing the surveillance and analyze the network behavior in multiple attack conditions. This work can be extended to design a robust multi-layer and multi-attack defense mechanism.

### *7.1.6 Section Summary*

Section 7.1 provides future research directions in securing SDN against LFAs. Since LFA is a novel security threat, there is a lack of available solutions which safeguard SDN against LFAs. In this regard, we highlighted the requirements for an optimal solution against LFAs in SDN. Finally, we discussed the need for proactive solutions which work on the real-time speed to effectively defend the LFAs.

As the current network technologies are becoming more and more virtualized, SDN is considered as the core building framework for those networks. Though this study provides a comprehensive solution against LFA, the attacks on control channel of SDN still needs to be identified in a comprehensive way. The centralized control in SDN sometimes becomes a point of bottleneck where the whole network remains under the availability of the controller which can be attacked any time. The in-band control strategy is cost-efficient in maintaining large data centers. However, it suffers from the problems of scalability and single point of failure.

The security solutions for the in-band controllers can be provided to handle the problems of the security. Besides the challenges at the controller, many other vulnerabilities are present at the data plane of SDN also. The TCAM memory at the data plane switches is used to store the flow rules in the SDN. However, TCAM memory induces a huge cost therefore, only a meager capacity is available in the current SDN-based infrastructure. The data plane switches possess a very low memory which can be overloaded with a variety of attacks. A flood of new flow rule installation requests can be instigated by the attacker which can deplete the TCAM memory and the switch may become in-accessible which can disrupt the communication. A carefully crafted attack on an ingress switch can cause the discrepancies in the network and badly affect the inflow of the traffic. Moreover, a variety of other attacks are possible in SDN, where a holistic security solution is required to mitigate such attacks.

In the current literature, there are many security solutions that address an individual aspect of the defense either control layer, data layer, or any individual channel. Therefore, a comprehensive solution to mitigate these attacks is necessary which can deal with different types of threats. Moreover, proactive solutions are required with a lower computational overhead as the available solutions pose a higher amount computational cost.

# REFERENCES

[1] V. K. Dimitrios Gkounis, Christos Liaskos, Xenofontas Dimitropoulos, , "On the interplay of link-flooding attacks and traffic engineering," *ACM SIGCOMM Computer Communication Review,* vol. 46, pp. 5-11, 2016.

[2] M. S. Kang, V. D. Gligor, and V. Sekar, "SPIFFY: Inducing Cost-Detectability Tradeoffs for Persistent Link-Flooding Attacks," in *Proc. Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, 2016, pp. 1-16.

[3] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *Proc. IEEE Symposium on Security and Privacy (SP)*, San Francisco, California, USA, 2013, pp. 127-141.

[4] A. Studer and A. Perrig, "The coremelt attack," in *Proc. Springer, European Symposium on Research in Computer Security (ESORICS)*, Saint-Malo, France, 2009, pp. 37-52.

[5] G. G. Alan Mclean, Archie Tse. (2013, June 24). *How the Cyberattack on Spamhaus Unfolded*. Available: https://archive.nytimes.com/www.nytimes.com/interactive/2013/03/30/technology/how-the-cyberattack-on-spamhaus-unfolded.html?

[6] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Communications Surveys & Tutorials*, vol. 16, pp. 2181-2206, 2014.

[7] 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 NetSoft Conference and Workshops (NetSoft)*, Seoul, South Korea, 2016, pp. 68-72.

[8] R. Mohammadi and R. Javidan, "An adaptive type-2 fuzzy traffic engineering method for video surveillance systems over software defined networks," *Multimedia Tools and Applications*, vol. 76, pp. 23627-23642, 2017.

[9] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. of the IEEE,* vol. 103, pp. 14-76, 2015.

[10] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *Proc. IEEE Int. Conf. on Computer Communications (INFOCOM)*, Turin, Italy, 2013, pp. 2211-2219.

[11] A. Yaar, A. Perrig, and D. Song, "SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks," in *Proc. IEEE Symposium on Security and Privacy*, California, USA, 2004, pp. 130-143.

[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, *et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 69-74, 2008.

[13] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys & Tutorials*, vol. 18, pp. 623-654, 2016.

[14] L. Wei and C. Fung, "FlowRanger: A request prioritizing algorithm for controller DoS attacks in software defined networks," in *Proc. IEEE Int. Conf. on Communications (ICC)*, London, UK, 2015, pp. 5254-5259.

[15] D. Smyth, S. McSweeney, D. O'Shea, and V. Cionca, "Detecting Link Fabrication Attacks in Software-Defined Networks," in *Proc. IEEE 26th Int. Conf. on Computer Communication and Networks (ICCCN)*, Vancouver, BC, Canada, 2017, pp. 1-8.

[16] L. Xue, X. Ma, X. Luo, E. W. Chan, T. T. Miu, and G. Gu, "LinkScope: Toward Detecting Target Link Flooding Attacks," *IEEE Transactions on Information Forensics and Security,* vol. 13, pp. 2423-2438, 2018.

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

[18] Q. A Wang, F. A Xiao, M. A Zhou, Z. A Wang, and H. A Ding, "Mitigating Link-flooding Attacks With Active Link Obfuscation," *Computing Research Repository (CoRR),* vol. abs/1703.09521, pp. 1-14, 2017.

[19] S. S. Jinwoo Kim, "Software-Defined HoneyNet:Towards Mitigating Link Flooding Attacks," in *Proc. 47th Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks Workshops*, Denver, CO, USA, 2017, pp. 99-100.

[20] S. B. Lee, M. S. Kang, and V. D. Gligor, "CoDef: Collaborative defense against large-scale link-flooding attacks," in *Proc. 9th ACM Conf. on Emerging Networking Experiments and Technologies (CoNEXT)*, Santa Barbara, California, USA 2013, pp. 417-428.

[21] C. Liaskos, V. Kotronis, and X. Dimitropoulos, "A novel framework for modeling and mitigating distributed link flooding attacks," in *Proc. IEEE 35th International Conference on Computer Communications (INFOCOM)*, San Francisco, CA, USA, 2016, pp. 1-9.

[22] T. Hirayama, K. Toyoda, and I. Sasase, "Fast target link flooding attack detection scheme by analyzing traceroute packets flow," in *Proc. IEEE Int. Workshop on Information Forensics and Security (WIFS)*, Rome, Italy, 2015, pp. 1-6.

[23] L. Xue, X. Luo, E. W. Chan, and X. Zhan, "Towards Detecting Target Link Flooding Attack," in *Proc. USENIX 28th Conf. on Large Installation System Administration (LISA)*, Seattle, WA, USA, 2014, pp. 81-96.

[24] D. Gkounis, V. Kotronis, C. Liaskos, and X. Dimitropoulos, "On the interplay of link-flooding attacks and traffic engineering," *ACM SIGCOMM Computer Communication Review,* vol. 46, pp. 5-11, 2016.

[25] L. Wang, Q. Li, Y. Jiang, and J. Wu, "Towards mitigating Link Flooding Attack via incremental SDN deployment," in *Proc. of IEEE Symposium on Computers and Communication (ISCC)*, Messina, Italy, 2016, pp. 397-402.

[26] A. Aydeger, N. Saputro, K. Akkaya, and M. Rahman, "Mitigating crossfire attacks using sdn-based moving target defense," in *Proc. of IEEE Conference on Local Computer Networks (LCN)*, Dubai, United Arab Emirates, 2016, pp. 627-630.

[27] D. M. Kristian Slavov, Makan Pourzandi, "Identifying and Addressing Vulnerabilities and Security Issues in SDN," Ericsson, Stockholm, Sweden2015.

[28] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *IEEE communications surveys & tutorials,* vol. 16, pp. 1955-1980, 2014.

[29] W. Zhou, L. Li, M. Luo, and W. Chou, "REST API design patterns for SDN northbound API," in *In Proc. IEEE 28th Int. Conf. on Advanced Information Networking and Applications Workshops (WAINA)*, Victoria, Canada, 2014, pp. 358-365.

[30] P. Sharma, S. Banerjee, S. Tandel, R. Aguiar, R. Amorim, and D. Pinheiro, "Enhancing network management frameworks with SDN-like control," in *Proc. IFIP/IEEE Int. Symposium on Integrated Network Management (IM)*, Ghent, Belgium, 2013, pp. 688-691.

[31] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future internet," *Computer Networks,* vol. 75, pp. 453-471, 2014.

[32] S. Cho, S. Chung, W. Lee, I. Joe, J. Park, S. Lee*, et al.*, "An software defined networking architecture design based on topic learning-enabled data distribution service middleware," *Advanced Science Letters,* vol. 21, pp. 461-464, 2015.

[33] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan*, et al.*, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Communications Magazine,* vol. 51, pp. 36-43, 2013.

[34] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," in *Proc. ACM 2nd SIGCOMM workshop on Hot topics in software defined networking (HotSDN)*, Hong Kong, China, 2013, pp. 7-12.

[35] H. W. Peng Zhang, Chengchen Hu, and Chuang Lin, "On Denial of Service Attacks in Software Defined Networks," *IEEE Network*, vol. 30, pp. 28-34, 2016.

[36] F. M. V. R. Diego Kreutz, Paulo Verissimo, "Towards Secure and Dependable Software-Defined Networks," in *Proc. ACM 2nd SIGCOMM workshop on Hot topics in software defined networking (HotSDN)*, Hong Kong, China, 2013, pp. 55-60

[37] G. B. Rishikesh Sahay, Zonghua Zhangy and Herv´e Debar, "Towards Autonomic DDoS Mitigation using Software Defined Networking

" in *Proc. NDSS Workshop on Security of Emerging Networking Technologies (SENT)*, San Diego, California, US, 2015, pp. 1-7.

[38] M. F. Z. Lobna Dridi, "SDN-Guard: DoS Attacks Mitigation in SDN Networks," in *Proc. IEEE 5th Int. Conf. on Cloud Networking*, PISA, Italy, 2016, pp. 212-217.

[39] K. K. G. G. F. Alagoz, "Defense Mechanisms against DDoS Attacks in SDN Environment," *IEEE Communications Magazine* vol. 55, pp. 175 - 179, 2017.

[40] B. a. T. Chandrasekaran, Brendan and Benson, Theophilus, "Isolating and Tolerating SDN Application Failures with LegoSDN," in *Proc. ACM Symposium on SDN Research (SOSR)*, Santa Clara, CA, USA, 2016, pp. 7:1--7:12.

[41] T. K. Karim ElDefrawy, "Byzantine Fault Tolerant Software-Defined Networking (SDN) Controllers," in *Proc. IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, Atlanta, GA, USA, 2016, pp. 208-213.

[42] H. Z. Naga Katta, Michael Freedman, Jennifer Rexford, "Ravana: Controller Fault-Tolerance in Software-Defined Networking," in *Proc. ACM 1st SIGCOMM Symposium on Software Defined Networking Research (SOSR)*, Santa Clara, California.

[43] J. R. S. Hyojoon Kim, Yoshio Turner, "CORONET: Fault Tolerance for Software Defined Networks," in *Proc. IEEE 20th Int. Conf. on Network Protocols (ICNP)*, Austin, TX, USA, 2012.

[44] A. R. J. Kuan-yin Chen, Ishant Kumar Siddhrau, "SDNShield: Towards more comprehensive defense against DDoS attacks on SDN control plane," in *Proc. IEEE Conf. on Communications and Network Security (CNS)*, Philadelphia, PA, USA, 2016.

[45] J. H. Lei Xu, Sungmin Hong, Jialong Zhang, Guofei Gu, "Attacking the Brain: Races in the SDN Control Plane," in *Proc. 26th USENIX Security Symposium (SEC)*, Vancouver, BC, Canada, 2017, pp. 451-468

[46] D. Z. Bin Yuan, Shui Yu, Senior Member, IEEE, Hai Jin, Senior Member, IEEE, Weizhong Qiang, and Jinan Shen, "Defending against Flow Table Overloading Attack in Software-Defined Networks," *Transactions on Services Computing,* vol. 12, pp. 231 - 246, 2019.

[47] C. P. akayuki Sasaki , Taeho Lee, Torsten Hoefler, Adrian Perrig, "SDNsec: Forwarding Accountability for the SDN Data Plane," in *Proc. IEEE Int. Conf. on Computer Communication and Networks (ICCCN)*, Waikoloa, HI, USA, 2016, pp. 1-10.

[48] S. Khan, A. Gani, A. W. A. Wahab, M. Guizani, and M. K. Khan, "Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art," *IEEE Communications Surveys & Tutorials,* vol. 19, pp. 303-324, 2017.

[49] K. S. S. Y. Y. Xiang, "Performance Analysis of Software-Defined Network Switch using M/Geo/1 Model," *IEEE Communications Letters,* vol. 20, pp. 2522-2525, September 2016 2016.

[50] Q. Niyaz, W. Sun, and M. Alam, "Impact on SDN Powered Network Services Under Adversarial Attacks," in *Proc. Intl. Conf. on Soft Computing and Software Engineering (SCSE)*, University of California, Berkeley, Suta, 2015, pp. 228-235.

[51] J. Wang, R. Wen, J. Li, F. Yan, B. Zhao, and F. Yu, "Detecting and Mitigating Target Link-Flooding Attacks Using SDN," *IEEE Transactions on Dependable and Secure Computing,* pp. 1-1, 2018.

[52] Q. Wang, F. Xiao, M. Zhou, Z. Wang, and H. Ding, "Mitigating Link-flooding Attacks With Active Link Obfuscation," *arXiv preprint arXiv:1703.09521,* 2017.

[53] C. Liaskos, V. Kotronis, and X. Dimitropoulos, "A novel framework for modeling and mitigating distributed link flooding attacks," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*, 2016, pp. 1-9.

[54] 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 Conference on Computer Communications (INFOCOM)*, San Francisco, CA, USA, 2015, pp. 729-737.

[55] 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/I SPA*, Tianjin, China, 2016, pp. 1008-1015.

[56] R. U. Rasool, M. Najam, H. F. Ahmad, W. Hua, and Z. Anwar, "A novel JSON based regular expression language for pattern matching in the internet of things," *Journal of Ambient Intelligence & Humanized Computing,* vol. 10, pp. 1463–1481, 2018.

[57] F. Yu, Z. Chen, Y. Diao, T. Lakshman, and R. H. Katz, "Fast and memory-efficient regular expression matching for deep packet inspection," in *Prof. ACM/IEEE Symposium on Architecture for Networking and Communications systems*, 2006, pp. 93-102.

[58] M. Becchi and P. Crowley, "A hybrid finite automaton for practical deep packet inspection," in *Proc. ACM CoNEXT Conference*, New York, USA, 2007, pp. 1-12.

[59] M. Najam, U. Younis, and R. ur Rasool, "Speculative parallel pattern matching using stride-k DFA for deep packet inspection," *Journal of Network and Computer Applications*, vol. 54, pp. 78-87, 2015.

[60] D. Ficara, A. Di Pietro, S. Giordano, G. Procissi, F. Vitucci, and G. Antichi, "Differential encoding of DFAs for fast regular expression matching," *IEEE/ACM Transactions On Networking*, vol. 19, pp. 683-694, 2011.

[61] T. Liu, Y. Yang, Y. Liu, Y. Sun, and L. Guo, "An efficient regular expressions compression algorithm from a new perspective," in *Proc. Int. Conf. on Computer Communications (INFOCOM)*, Shanghai, China, 2011, pp. 2129-2137.

[62] D. Luchaup, R. Smith, C. Estan, and S. Jha, "Multi-byte regular expression matching with speculation," in *Proc. Springer Int. Workshop on Recent Advances in Intrusion Detection (RAID)*, Saint-Malo, France, 2009, pp. 284-303.

[63] G. Vasiliadis, M. Polychronakis, S. Antonatos, E. P. Markatos, and S. Ioannidis, "Regular expression matching on graphics hardware for intrusion detection," in *International Workshop on Recent Advances in Intrusion Detection*, 2009, pp. 265-283.

[64] S. Kong, R. Smith, and C. Estan, "Efficient signature matching with multiple alphabet compression tables," in *Proc. ACM 4th international Conf. on Security and privacy in communication netowrks (SecureComm)*, Istanbul, Turkey, 2008, pp. 1-10.

[65] D. Ficara, S. Giordano, G. Procissi, F. Vitucci, G. Antichi, and A. Di Pietro, "An improved DFA for fast regular expression matching," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 29-40, 2008.

[66] M. Becchi and S. Cadambi, "Memory-efficient regular expression search using state merging," in *Proc. 26th IEEE Int. Conf. on Computer Communications (INFOCOM)*, New York, NY, USA, 2007, pp. 1064-1072.

[67] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," in *ACM SIGCOMM Computer Communication Review*, 2006, pp. 339-350.

[68] B. Heller. (2018, June 24). *A Virtual Network on Your Desktop. An instant virtual network on your laptop (or other pc)-mininet*

Available: http://mininet.org/

[69] N. Mate. (2018). *Project Floodlight*. Available: http://www.projectfloodlight.org/floodlight/

[70] M. Karakus and A. Durresi, "Service Cost in Software Defined Networking (SDN)," in *Proc. IEEE Int. Conf. on Advanced Information Networking & Applications (AINA)*, Taipei, Taiwan, 2017, pp. 468-475.

[71] M. Canini, P. Kuznetsov, D. Levin, and S. Schmid, "A distributed and robust sdn control plane for transactional network updates," in *Proc. IEEE Int. Conf Computer Communications (INFOCOM)*, Kowloon, Hong Kong, 2015, pp. 190-198.

[72] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE communications surveys & tutorials*, vol. 15, pp. 2046-2069, 2013.

[73] M. S. Kang and V. D. Gligor, "Routing bottlenecks in the internet: Causes, exploits, and countermeasures," in *Proc. ACM SIGSAC Conf. on Computer and Communications Security*, Scottsdale, Arizona, USA, 2014, pp. 321-333.

[74] D. Setsirichok, T. Piroonratana, W. Wongseree, T. Usavanarong, N. Paulkhaolarn, C. Kanjanakorn, *et al.*, "Classification of complete blood count and haemoglobin typing data by a C4.5 decision tree, a naïve Bayes classifier and a multilayer perceptron for thalassaemia screening," *Biomedical Signal Processing & Control*, vol. 7, pp. 202-212, 2012.

[75] 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. on Computer Communications (INFOCOM)*, Atlanta, GA, USA, 2017, pp. 1-9.

[76] M. Zhang, J. Bi, J. Bai, and G. Li, "FloodShield: Securing the SDN Infrastructure Against Denial-of-Service Attacks," in *Proc. IEEE 17th Int. Conf. On Trust, Security And Privacy In Computing and Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, New York, NY, USA, 2018, pp. 687-698.

[77] D. Kobialka. (2018, June 24). *Kaspersky Lab Study: Average Cost of Enterprise DDoS Attack Totals $2M*. Available: https://www.msspalert.com/cybersecurity-research/kaspersky-lab-study-average-cost-of-enterprise-ddos-attack-totals-2m/

[78] R. U. Rasool, W. Hua, W. Rafique, J. Yong, and J. Cao, "A Study on Securing Software Defined Networks," in *Proc. Springer Int. Conf. on Web Information Systems Engineering (WISE)*, Moscow, Russia, 2017, pp. 479-489.

[79] H. W. R. U. Rasool, U. Ashraf, K. Ahmed, Z. Anwar, W. Rafique, "A Survey of Link Flooding Attacks in Software Defined Network Ecosystems," *ACM Computing Surveys*, 2019, Under Review.

[80] S. Khan, A. Gani, A. W. A. Wahab, A. Abdelaziz, and M. A. Bagiwa, "FML: A novel forensics management layer for software defined networks," *Proc. IEEE 6th Int. Conf. on Cloud System and Big Data Engineering (Confluence)*, vol. 44, pp. 619-623, 2016.

[81] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, pp. 1617-1634, 2014.

[82] Y. Cai, F. R. Yu, C. Liang, B. Sun, and Q. Yan, "Software-defined device-to-device (D2D) communications in virtual wireless networks with imperfect network state information (NSI)," *IEEE Transactions on Vehicular Technology*, vol. 65, pp. 7349-7360, 2016.

[83] A. Csoma, B. Sonkoly, L. Csikor, F. Németh, A. Gulyas, W. Tavernier*, et al.*, "ESCAPE: Extensible service chain prototyping environment using mininet, click, netconf and pox," in *ACM SIGCOMM Computer Communication Review*, 2014, pp. 125-126.

[84] M. Brandt, R. Khondoker, R. Marx, and K. Bayarou, "Security Analysis of Software Defined Networking Protocols—OpenFlow, OF-Config and OVSDB," in *Proc. IEEE 5th Int. Conf. on Communications and Electronics (ICCE)*, Da Nang, Vietnam 2014, pp. 51-56.

[85] Y. Qian, W. You, and K. Qian, "OpenFlow flow table overflow attacks and countermeasures," in *Proc. IEEE European Conf.  on Networks and Communications (EuCNC)*, Athens, Greece, 2016, pp. 205-209.

[86] T. Xu, D. Gao, P. Dong, C. H. Foh, and H. Zhang, "Mitigating the Table-Overflow Attack in Software-Defined Networking," *IEEE Transactions on Network and Service Management*, vol. 14, pp. 1086-1097, 2017.

[87] L. Zhang, S. Wang, S. Xu, R. Lin, and H. Yu, "TimeoutX: An adaptive flow table management method in software defined networks," in *Proc. Global Communications Conference (GLOBECOM)*, San Diego, CA, USA, 2015, pp. 1-6.

[88] B. Leng, L. Huang, X. Wang, H. Xu, and Y. Zhang, "A mechanism for reducing flow tables in software defined network," in *Proc. IEEE Int. Conf. on Communications (ICC)*, London, UK, 2015, pp. 5302-5307.

[89] D. B. Rawat and S. R. Reddy, "Software defined networking architecture, security and energy efficiency: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, pp. 325-346, 2017.

[90] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, pp. 2317-2346, 2015.

[91] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Computer Networks*, vol. 62, pp. 122-136, 2014.

[92] R. Kandoi and M. Antikainen, "Denial-of-service attacks in OpenFlow SDN networks," in *Proc. IFIP/IEEE Int. Symposium on Integrated Network Management (IM)*, Ottawa, ON, Canada, 2015, pp. 1322-1326.

[93] H. T. N. Tri and K. Kim, "Assessing the impact of resource attack in Software Defined Network," in *Proc. IEEE Int. Conf. on Information Networking (ICOIN)*, Combodia, 2015, pp. 420-425.

[94] X. W. Arjun P. Athreya, Yu Seung Kim, Yuan Tian and Patrick Tague, "Resistance is Not Futile: Detecting DDoS Attacks without Packet Inspection," in *Proc. 14th Int. Workshop on Information Security Applications*, Jeju Island, Korea, 2013, pp. 1-15.

[95] W. H. Highleyman. (2013) History's Largest DDoS Attack. *Availability Digest*. 1-5. Available: http://www.availabilitydigest.com/public_articles/0804/spamhaus.pdf

[96] J. M. Vidal, A. L. S. Orozco, and L. J. G. Villalba, "Adaptive artificial immune networks for mitigating DoS flooding attacks," *Swarm and Evolutionary Computation*, vol. 38, pp. 94 -108, 2018.

[97] H. Luo, Z. Chen, J. Li, and A. V. Vasilakos, "Preventing distributed denial-of-service flooding attacks with dynamic path identifiers," *IEEE Transactions on Information Forensics and Security*, vol. 12, pp. 1801-1815, 2017.

[98] D. Gkounis, "Cross-domain DoS link-flooding attack detection and mitigation using SDN principles," Masters Thesis, Information Technology and Electrical Engineering, ETH Zurich, Zurich, Switzerland, 2014.

[99] R. Mohammadi, R. Javidan, and M. Conti, "Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks," *IEEE Transactions on Network and Service Management*, vol. 14, pp. 487-497, 2017.

[100] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Computer Networks*, vol. 112, pp. 279-293, 2017.

[101] F. J. Ryba, M. Orlinski, M. Wählisch, C. Rossow, and T. C. Schmidt, "Amplification and DRDoS Attack Defense-A Survey and New Perspectives," *arXiv preprint*, vol. arXiv:1505.07892, pp. 1-19, 2015.

[102] R. Sahay, G. Blanc, Z. Zhang, K. Toumi, and H. Debar, "Adaptive policy-driven attack mitigation in SDN," in *Proc. ACM 1st Int. Workshop on Security and Dependability of Multi-Domain Infrastructures*, Belgrade, Serbia, 2017, pp. 4:1--4:6.

[103] 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. Springer Int. Conf. on 5G for Future Wireless Networks*, Beijing, China, 2017, pp. 371-384.

[104] K. S. Hoon, K. C. Yeo, S. Azam, B. Shunmugam, and F. De Boer, "Critical review of machine learning approaches to apply big data analytics in DDoS forensics," in *Proc. IEEE Intl. Conf. on Computer Communication and Informatics (ICCCI)*, Coimbatore, India, 2018, pp. 1-5.

[105] B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity," in *Proc. ACM 23rd Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, 2017, pp. 1723-1732.

[106] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing, In Press*, pp. 1-13, 2017.

[107] M. Agarwal, D. Pasumarthi, S. Biswas, and S. Nandi, "Machine learning approach for detection of flooding DoS attacks in 802.11 networks and attacker localization," *International Journal of Machine Learning and Cybernetics*, vol. 7, pp. 1035-1051, 2016.

[108] A. Kalliola, K. Lee, H. Lee, and T. Aura, "Flooding DDoS mitigation and traffic management with software defined networking," in *Proc. IEEE Int. Conf. on Cloud Networking (CloudNet)*, Niagara Falls, Canada, 2015, pp. 248-254.

[109] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Computing*, vol. 20, pp. 343-357, 2016.

[110] Z. He, T. Zhang, and R. B. Lee, "Machine learning based ddos attack detection from source side in cloud," in *Proc. IEEE 4th Int. Conf. on Cyber Security and Cloud Computing (CSCloud)*, New York, USA, 2017, pp. 114-120.

[111] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, "SSHCure: a flow-based SSH intrusion detection system," in *Proc. Springer 6th Int. Conf. on Autonomous Infrastructure, Management, and Security (AIMS)* luxembourg, luxembourg 2012, pp. 86-97.

[112] L. R. Knudsen and M. J. Robshaw, *Brute force attacks in the series of The Block Cipher Companion. Information Security and Cryptography*. Berlin, Heidelberg: Springer, 2001.

[113] R. Daş, A. Karabade, and G. Tuna, "Common network attack types and defense mechanisms," in *Proc. IEEE 23rd Conf. on Signal Processing and Communications Applications (SIU)*, Malatya, Turkey, 2015, pp. 2658-2661.

[114] D. H. Ahmed, M. Hussin, A. Abdullah, and R. A. R. Mahmood, "Distributed Defense Scheme for Managing DNS Reflection Attack in Network Communication Systems," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 8, pp. 71-75, 2016.

[115] X. N. Zeng, A. Ghrayeb, and M. Hasna, "Joint Optimal Threshold-Based Relaying and ML Detection in Network-Coded Two-Way Relay Channels," *IEEE Transactions on Communications,* vol. 60, pp. 2657-2667, 2012.

[116] G. A. Ajaeiya, N. Adalian, I. H. Elhajj, A. Kayssi, and A. Chehab, "Flow-based Intrusion Detection System for SDN," in *Proc. IEEE Symposium on Computers and Communications (ISCC),* Heraklion, Greece, 2017, pp. 787-793.

[117] C. J. Bernardos, A. De La Oliva, P. Serrano, A. Banchs, L. M. Contreras, H. Jin, *et al.*, "An architecture for software defined wireless networking," *IEEE wireless communications,* vol. 21, pp. 52-61, 2014.

[118] S. Shin, P. A. Porras, V. Yegneswaran, M. W. Fong, G. Gu, and M. Tyson, "FRESCO: Modular Composable Security Services for Software-Defined Networks," in *Proc. 20th Annual Network & Distributed System Security Symposium*, San Diego, CA United States, 2013, pp. 1-16.

[119] J. Schulz-Zander, C. Mayer, B. Ciobotaru, S. Schmid, and A. Feldmann, "Unified Programmability of Virtualized Network Functions and Software-Defined Wireless Networks," *IEEE Transactions on Network and Service Management,* vol. 14, pp. 1046-1060, 2017.

[120] H. Wang, H. Tang, and S. Zhang, "Joint Optimization in Software Defined Wireless Networks with Network Coded Opportunistic Routing," in *Proc. IEEE 14th Int. Conf. on Mobile Ad Hoc and Sensor Systems (MASS),* Orlando, FL, USA, 2017, pp. 298-302.

[121] C.-F. Liu, S. Samarakoon, M. Bennis, and H. V. Poor, "Fronthaul-Aware Software-Defined Wireless Networks: Resource Allocation and User Scheduling," *IEEE Transactions on Wireless Communications,* vol. 17, pp. 533-547, 2018.

[122] C. Liang, Y. He, F. R. Yu, and N. Zhao, "Enhancing QoE-Aware Wireless Edge Caching With Software-Defined Wireless Networks," *IEEE Transactions on Wireless Communications,* vol. 16, pp. 6912-6925, 2017.

[123] P. Graubner, M. Sommer, M. Hollick, and B. Freisleben, "Dynamic role assignment in Software-Defined Wireless Networks," in *Proc. IEEE Symposium on Computers and Communications (ISCC),* Heraklion, Greece, 2017, pp. 760-766.

[124] B. O. Kahjogh and G. Bernstein, "Energy and latency optimization in software defined wireless networks," in *Proc. IEEE Int. Conf. on Ubiquitous and Future Networks (ICUFN)*, Milan, Italy, 2017, pp. 714-719.

[125] X. Liu, A. Liu, and Z. Li, "Adaptive Broadcast Times for Program Codes in Software Defined Wireless Networks," in *Proc. IEEE Int. Conf. Mobile Ad-Hoc and Sensor Networks (MSN)*, Hefei, China, 2016, pp. 405-408.

[126] M. J. Abdel-Rahman, E. A. Mazied, A. MacKenzie, S. Midkiff, M. R. Rizk, and M. El-Nainay, "On stochastic controller placement in software-defined wireless networks," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, San Fransisco, CA, USA, 2017, pp. 1-6.

[127] K. Mizuyama, Y. Taenaka, and K. Tsukamoto, "Estimation based adaptable Flow Aggregation Method for reducing control traffic on Software Defined wireless Networks," in *Proc. IEEE Int. Conf. on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Kona, HI, USA, 2017, pp. 363-368.

[128] B. Cao, Y. Li, C. Wang, G. Feng, S. Qin, and Y. Zhou, "Resource Allocation in Software Defined Wireless Networks," *IEEE Network,* vol. 31, pp. 44-51, 2017.

[129] S. M. Mingjie FENG, Tao JIANG, "Enhancing the performance of future wireless networks with software-defined networking," *Frontiers of Information Technology & Electronic Engineering,* vol. 2016 17(7), pp. 606-619, 2016.

[130] N. Zhang, N. Cheng, N. Lu, H. Zhou, J. W. Mark, and X. Shen, "Risk-aware cooperative spectrum access for multi-channel cognitive radio networks," *IEEE Journal on Selected Areas in Communications,* vol. 32, pp. 516-527, 2014.

[131] Y. T. Hou, Y. Shi, and H. D. Sherali, "Optimal spectrum sharing for multi-hop software defined radio networks," in *Proc. IEEE 26th Int. Conf. on Computer Communications (INFOCOM),* Barcelona, Spain, 2007, pp. 1-9.

[132] I. Ahmad, S. N. Karunarathna, M. Ylianttila, and A. Gurtov, *Load balancing in software defined mobile networks*: John Wiley & Sons, Ltd., 2015.

[133] H. Selvi, S. Güner, G. Gür, and F. Alagöz, "The controller placement problem in software defined mobile networks (SDMN)," in *Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture*, ed: John Wiley & Sons, Ltd, 2015, pp. 129-147.

[134] R. G. L. Narayanan, *Software Defined Networks for Mobile Application Services*: John Wiley & Sons, Ltd, 2015.

[135] C. Liang and F. R. Yu, "Enhancing mobile edge caching with bandwidth provisioning in software-defined mobile networks," in *Proc. IEEE Int. Conf. on Communications (ICC)*, Paris, France, 2017, pp. 1-6.

[136] M. Liyanage, I. Ahmad, J. Okwuibe, M. Ylianttila, H. Kabir, J. L. Santos*, et al.*, "Enhancing Security of Software Defined Mobile Networks," *IEEE Access,* vol. 5, pp. 9422-9438, 2017.

[137] C. Liang and F. R. Yu, "Bandwidth Provisioning in Cache-Enabled Software-Defined Mobile Networks: A Robust Optimization Approach," in *Proc. IEEE 84th Vehicular Technology Conference (VTC)*, Montreal, QC, Canada, 2016, pp. 1-5.

[138] I. Ahmad, M. Liyanage, S. Namal, M. Ylianttila, A. Gurtov, M. Eckert*, et al.*, "New concepts for traffic, resource and mobility management in software-defined mobile networks," in *Proc. IEEE 12th Int. Conf. Wireless On-demand Network Systems and Services (WONS)*, Cortina d'Ampezzo, Italy, 2016, pp. 1-8.

[139] M. Liyanage, I. Ahmed, M. Ylianttila, J. L. Santos, R. Kantola, O. L. Perez*, et al.*, "Security for future software defined mobile networks," in *Proc. IEEE 9th Int. Conf. on Next Generation Mobile Applications, Services and Technologies*, Cambridge, UK, 2015, pp. 256-264.

[140] L. J. Chaves, V. M. Eichemberger, I. C. Garcia, and E. R. M. Madeira, "Integrating OpenFlow to LTE: Some issues toward software-defined mobile networks," in *Proc. IEEE Int. Conf. on New Technologies, Mobility and Security (NTMS)*, Paris, France, 2015, pp. 1-5.

[141] R. Riggio, M. K. Marina, and T. Rasheed, "Interference management in software-defined mobile networks," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, 2015, pp. 626-632.

[142] M. Liyanage, M. Ylianttila, and A. Gurtov, "Securing the control channel of software-defined mobile networks," in *Proc. IEEE 15th Int. Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Sydney, NSW, Australia, 2014, pp. 1-6.

[143] I. Katzela and M. Naghshineh, "Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey," *IEEE personal communications,* vol. 3, pp. 10-31, 1996.

[144] F.-Y. Wang, "Parallel control and management for intelligent transportation systems: Concepts, architectures, and applications," *IEEE Transactions on Intelligent Transportation Systems,* vol. 11, pp. 630-638, 2010.

[145] J. Chung, G. Gonzalez, I. Armuelles, T. Robles, R. Alcarria, and A. Morales, "Experiences and challenges in deploying openflow over real wireless mesh networks," *IEEE Latin America Transactions*, vol. 11, pp. 955-961, 2013.

[146] F. Yang, V. Gondi, J. O. Hallstrom, K.-C. Wang, and G. Eidson, "OpenFlow-based load balancing for wireless mesh infrastructure," in *Proc. IEEE Consumer Communications and Networking Conference (CCNC)*, Las Vegas, NV, USA, 2014, pp. 444-449.

[147] W. Zhao and J. Xie, "IMeX: intergateway cross-layer handoffs in internet-based infrastructure wireless mesh networks," *IEEE Transactions on Mobile Computing*, vol. 11, pp. 1585-1600, 2012.

[148] M. K. Marina, S. R. Das, and A. P. Subramanian, "A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks," *Computer networks*, vol. 54, pp. 241-256, 2010.

[149] S. L. K. Sood, S. Yu, and Y. Xiang, "Dynamic Access Point Association Using Software-Defined Networking," in *IEEE Telecommunication Networks and Applications Conference (ITNAC)*, Sydney, NSW, Australia, 2015, pp. 226-231.

[150] P. Dely, A. Kassler, and N. Bayer, "Openflow for wireless mesh networks," in *Proc. IEEE Int. Conf. on Computer Communications and Networks (ICCCN)*, Maui, HI, USA, 2011, pp. 1-6.

[151] A. Detti, C. Pisa, S. Salsano, and N. Blefari-Melazzi, "Wireless mesh software defined networks (wmSDN)," in *Proc. IEEE 9th Int. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Lyon, France, 2013, pp. 89-95.

[152] H. Huang, P. Li, S. Guo, and W. Zhuang, "Software-defined wireless mesh networks: architecture and traffic orchestration," *IEEE network*, vol. 29, pp. 24-30, 2015.

[153] J. Chen, B. Liu, H. Zhou, Q. Yu, G. Lin, and X. Shen, "QoS-Driven Efficient Client Association in High-Density Software Defined WLAN," *IEEE Transactions on Vehicular Technology*, vol. 66, pp. 7372 - 7383, 2017.

[154] A. Amelyanovich, M. Shpakov, A. Muthanna, M. Buinevich, and A. Vladyko, "Centralized control of traffic flows in wireless LANs based on the SDN concept," in *Proc. IEEE Systems of Signal Synchronization, Generating and Processing in Telecommunications (SINKHROINFO)*, Kazan, Russia, 2017, pp. 1-5.

[155] X. Sang, Q. Wu, and H. Li, "Client-network collaborative load balancing mechanism for WLAN based on SDN and 802.11 u," in *Proc. IEEE 13th Int. Wireless Communications and Mobile Computing Conference (IWCMC)*, Valencia, Spain, 2017, pp. 506-511.

[156] C. C. Min Cheng Chan, Jun-Xian Huwang, Chien Chao Tseng, "OpenNet: A simulator for software-defined wireless local area network," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, Istanbul, Turkey, 2014, pp. 3332-3336.

[157] S. Shin and G. Gu, "Attacking software-defined networks: A first feasibility study," in *Proc. ACM 2nd SIGCOMM workshop on Hot topics in software defined networking*, Hong Kong, China, 2013, pp. 165-166.

[158] M. Antikainen, T. Aura, and M. Särelä, "Spook in your network: Attacking an sdn with a compromised openflow switch," in *Proc. Springer Nordic Conference on Secure IT Systems*, Tromsø, Norway, 2014, pp. 229-244.

[159] J. Leng, Y. Zhou, J. Zhang, and C. Hu, "An inference attack model for flow table capacity and usage: Exploiting the vulnerability of flow table overflow in software-defined network," *arXiv preprint,* vol. Preprint arXiv:1504.03095, pp. 1-13, 2015.

[160] H. Wang, L. Xu, and G. Gu, "OF-GUARD: A DoS attack prevention extension in software-defined networks," *The Open Network Summit (ONS)*, 2014.

[161] T. Chin, X. Mountrouidou, X. Li, and K. Xiong, "Selective packet inspection to detect DoS flooding using software defined networking (SDN)," in *Proc. IEEE 35th Int. Conf. on Distributed Computing Systems Workshops (ICDCSW)*, Columbus, OH, USA, 2015, pp. 95-99.

[162] N.-N. Dao, J. Park, M. Park, and S. Cho, "A feasible method to combat against DDoS attack in SDN network," in *Proc. IEEE Int. Conf. on Information Networking (ICOIN)*, Cambodia, Cambodia, 2015, pp. 309-311.

[163] M. C. M. Ambrosin, F. De Gaspari, R. Poovendran, "LineSwitch: Tackling Control Plane Saturation Attacks in Software-Defined Networking," *IEEE/ACM Transactions on Networking,* vol. Volume: 25, Issue: 2, April 2017, p. 14, 29 November 2016.

[164] H. Wang, L. Xu, and G. Gu, "Floodguard: A dos attack prevention extension in software-defined networks," in *Proc. IEEE 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2015, pp. 239-250.

[165] G. Waller. (2015, June 24). *Gartner Says a Massive Shift to Hybrid Infrastructure Services Is Underway Gartner Survey*. Available: http://www.gartner.com/newsroom/id/3666917

[166] F. Kobuszewski. (2017, June 24). *Software Defined Networking: Trend or technology movement? Will SDN be coming soon to a network near you?* Available: http://www.networkworld.com/article/2981667/cisco-subnet/software-defined-networking-trend-or-technology-movement.html

[167] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Proc. Springer Intl. Workshop on Recent Advances in Intrusion Detection*, 2011, pp. 161-180.

[168] S. Lim, J. Ha, H. Kim, Y. Kim, and S. Yang, "A SDN-oriented DDoS blocking scheme for botnet-based attacks," in *Proc. IEEE 6th Int. Conf. on Ubiquitous and Future Networks (ICUFN)*, Shanghai, China, 2014, pp. 63-68.

[169] Q. Yan and F. R. Yu, "Distributed denial of service attacks in software-defined networking with cloud computing," *IEEE Communications Magazine*, vol. 53, pp. 52-59, 2015.

[170] A. Akhunzada, A. Gani, N. B. Anuar, A. Abdelaziz, M. K. Khan, A. Hayat, *et al.*, "Secure and dependable software defined networks," *Journal of Network and Computer Applications*, vol. 61, pp. 199-221, 2016.

[171] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Communications Surveys & Tutorials*, vol. 18, pp. 655-685, 2016.

[172] J. A. S. Narmeen Zakaria Bawany, Khaled Salah, "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions," *Arab J Sci Eng*, vol. 42, pp. 425–441, 2017.

[173] J. Fruhlinger. (2018, June 24). *What is WannaCry ransomware, how does it infect, and who was responsible?* Available: https://www.csoonline.com/article/3227906/what-is-wannacry-ransomware-how-does-it-infect-and-who-was-responsible.html

[174] B. Appleby. (2019, June 24). *Protect Your Business and Stay Online During a DDoS Attack*. Available: https://www.f5.com/pdf/products/f5-silverline-ddos-protection-datasheet.pdf

[175] R. U. Rasool, U. Ashraf, K. Ahmed, H. Wang, W. Rafique, and Z. Anwar, "Cyberpulse: A Machine Learning Based Link Flooding Attack Mitigation System for Software Defined Networks," *IEEE Access*, vol. 7, pp. 34885-34899, 2019.

[176] T. V. Phan, N. K. Bao, and M. Park, "Distributed-SOM: A novel performance bottleneck handler for large-sized software-defined networks under flooding attacks," *Journal of Network and Computer Applications*, vol. 91, pp. 14-25, 2017.

[177] D. Boro and D. K. Bhattacharyya, "DyProSD: a dynamic protocol specific defense for high-rate DDoS flooding attacks," *Microsystem Technologies*, vol. 23, pp. 593-611, 2017.

[178] T. V. Phan, T. Van Toan, D. Van Tuyen, T. T. Huong, and N. H. Thanh, "OpenFlowSIA: An optimized protection scheme for software-defined networks from flooding attacks," in

*Proc. IEEE 6th Int. Conf. on Communications and Electronics (ICCE)*, Ha Long, Vietnam, 2016, pp. 13-18.

[179] C. Li, J. Yang, Z. Wang, F. Li, and Y. Yang, "A Lightweight DDoS Flooding Attack Detection Algorithm Based on Synchronous Long Flows," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, San Diego, CA, USA, 2015, pp. 1-6.

[180] V. A. Foroushani and A. N. Zincir-Heywood, "TDFA: Traceback-based defense against DDoS flooding attacks," in *Poc. IEEE 28th Int. Conf. on Advanced Information Networking and Applications (AINA)*, Victoria, BC, Canada, 2014, pp. 597-604.

[181] H. W. Raihan Ur Rasool, Usman Ashraf, Zahid Anwar, Khandakar Ahmed, Wajid Rafique, "CyberPulse: A Security Framework for Link Flooding Attacks Mitigation in Software Defined Networks," *IEEE Transactions in Network and Service Management, Under Review*, pp. 1-14, 2019.

[182] T. Bray, "The javascript object notation (json) data interchange format," ed: Internet Engineering Task Force (IETF), 2017, p. 22.

[183] (2014, June 24). *OpenFlow Switch Specification V1. 4.0*. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf

[184] D. a. K. T. Dheeru, Efi, "UCI Machine Learning Repository," I. University of California, School of Information and Computer Sciences, Ed., April 20, 2018 ed. USA: University of California, 2017.

[185] (2018, June 24). *Network Resource Utilization*. Available: http://nsl.csie.nctu.edu.tw/nctuns.html

[186] H. P. Company. (1999, June 24). *Utilization, HP TopTools for Hubs & Switches*. Available: http://hp.com/rnd/device_help/help/hpwnd/webhelp/HPJ4093A/utilization.htm

[187] B. Commentz-Walter, "A string matching algorithm fast on the average," in *Proc. Int. Colloquium on Automata, Languages, and Programming*, Verlag London, UK, 1979, pp. 118-132.

[188] J. Kuri and G. Navarro, "A fast algorithm for multi-pattern searching," in *Proc. IEEE 7th Int. Symposium on String Processing and Information Retrieval (SPIRE)*, A Curuna, Spain, Spain, 2000, pp. 169-180.

[189] C. J. Coit, S. Staniford, and J. McAlerney, "Towards faster string matching for intrusion detection or exceeding the speed of snort," in *Proc. IEEE Information Survivability Conference & Exposition (DARPA)*, Anaheim, CA, USA, 2001, pp. 367-373.

[190] M. Fisk and G. Varghese, "Applying fast string matching to intrusion detection," United States. Department of Energy, Los Alamos National Laboratory, U.S2002.

[191] M. A. Sustik and J. Moore, "String searching over small alphabets," Report of Computer Science Department, University of Texas at Austin, Texas at Austin, USA2007.

[192] M. E. Nebel, "Fast string matching by using probabilities: on an optimal mismatch variant of Horspool's algorithm," *Theoretical Computer Science,* vol. 359, pp. 329-343, 2006.

[193] N. Tuck, T. Sherwood, B. Calder, and G. Varghese, "Deterministic memory-efficient string matching algorithms for intrusion detection," in *Proc. IEEE 23rd Int. Conf. on Computer Communications (INFOCOM)* 2004, pp. 2628-2639.

[194] D. P. Charboneau, "Apparatuses, systems, and methods for efficient graph pattern matching and querying," US Patent, 2013.

[195] P. Ravindra, H. Kim, and K. Anyanwu, "An intermediate algebra for optimizing RDF graph pattern matching on MapReduce," in *Peoc. Springer 8th Int. extended semantic web conf. on The semanic web: research and applications (ESWC),* Heraklion, Crete, Greece, 2011, pp. 46-61.

[196] B. Emir, "Object-oriented pattern matching," in *Proc. Springer European Conf. on Object-Oriented Programming (ECOOP)*, Berlin, Germany, 2007, pp. 273--298.

[197] M. Nourian, X. Wang, X. Yu, W.-c. Feng, and M. Becchi, "Demystifying automata processing: GPUs, FPGAs or Micron's AP?," in *Proc. ACM Int. Conf. on Supercomputing,* Chicago, IL, USA, 2017, pp. 1:1--1:11.

[198] M. Najam, U. Younis, and R. U. Rasool, "Multi-byte Pattern Matching Using Stride-K DFA for High Speed Deep Packet Inspection," in *Proc. IEEE Int. Conf. on Computational Science and Engineering (CSE)*, Chengdu, China, 2014, pp. 547-553.

[199] T. Liu, A. X. Liu, J. Shi, Y. Sun, and L. Guo, "Towards fast and optimal grouping of regular expressions via DFA size estimation," *IEEE Journal on Selected Areas in Communications,* vol. 32, pp. 1797-1809, 2014.

[200] Y.-K. Chang, Y.-S. Li, and Y.-T. Chen, "A Memory Efficient DFA Using Compression and Pattern Segmentation," *Procedia Computer Science,* vol. 56, pp. 292-299, 2015.

[201] R. Antonello, S. Fernandes, D. Sadok, J. Kelner, and G. Szabó, "Design and optimizations for efficient regular expression matching in DPI systems," *Computer Communications,* vol. 61, pp. 103-120, 2015.

[202] J. Patel, A. X. Liu, and E. Torng, "Bypassing space explosion in high-speed regular expression matching," *IEEE/ACM Transactions on Networking,* vol. 22, pp. 1701-1714, 2014.

[203] Y. Xu, J. Jiang, R. Wei, Y. Song, and H. J. Chao, "TFA: a tunable finite automaton for pattern matching in network intrusion detection systems," *IEEE journal on selected areas in communications,* vol. 32, pp. 1810-1821, 2014.

[204] X. Yu, B. Lin, and M. Becchi, "Revisiting state blow-up: Automatically building augmented-fa while preserving functional equivalence," *IEEE Journal on Selected Areas in Communications,* vol. 32, pp. 1822-1833, 2014.

[205] A. X. Chang and C. D. Manning, "TokensRegex: Defining cascaded regular expressions over tokens," Stanford University, CA, USA2014.

[206] Y. Fang, R. ur Rasool, D. Vasudevan, and A. A. Chien, "Generalized pattern matching micro-engine," in *Proc. 4th Workshop on Architectures and Systems for Big Data (ASBD), Held in conjunction with The 41st International Symposium on Computer Architecture (ISCA)*, Minneapolis, MN, USA, 2014, pp. 1-10.

[207] A. Giurca and E. Pascalau, "JSON Rules - The JavaScript Rule Engine," in *Proc. 4th Workshop on Knowledge Engineering and Software Engineering (KESE) at the 31st German Conference on Artificial Intelligence*, Kaiserslautern, Germany, 2008.

[208] D. Tomaszuk, "Document-oriented triple store based on RDF/JSON," *Studies in Logic, Grammar and Rhetoric,The Journal of University of Bialystok,* vol. 4, pp. 125-140, 2010.

[209] H.-J. Tsai, C.-C. Chen, Y.-C. Peng, Y.-H. Tsao, Y.-N. Chiang, W.-C. Zhao*, et al.*, "A Flexible Wildcard-Pattern Matching Accelerator via Simultaneous Discrete Finite Automata," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 25, pp. 3302-3316, 2017.

[210] K. Wang, Z. Fu, X. Hu, and J. Li, "Practical regular expression matching free of scalability and performance barriers," *Computer Communications,* vol. 54, pp. 97-119, 2014.

[211] K. Wang and J. Li, "Towards fast regular expression matching in practice," *ACM SIGCOMM Computer Communication Review,* vol. 43, pp. 531-532, 2013.

[212] N. Moreira and R. Reis, "Implementation and Application of Automata," in *Proc. Springer 17th Int. Conf. on Implementation and Application of Automata (CIAA)*, Implementation and Application of Automata:, 2012, pp. 1-381.

[213] M. Thottan, G. Liu, and C. Ji, "Anomaly detection approaches for communication networks," in *Algorithms for Next Generation Networks*, ed: Springer London, 2010, pp. 239-261.

[214] M. Becchi and P. Crowley, "Extending finite automata to efficiently match perl-compatible regular expressions," in *Proc. ACM CoNEXT Conference*, Madrid, Spain 2008, pp. 25:1--25:12.

[215] E. Lengyel. (2017, June 24). *OpenDDL, Open Data Description Language*

Available: Available from: http://openddl.org/.

[216] D. Wijnand. (2018, June 24). *HOCON, "Human-Optimized Config Object Notation"*. . Available: Available from: https://github.com/lightbend/config/blob/master/HOCON.md

[217] C. Evans. (2001, June 24). *YAML, "YAML Ain't Markup Language"*. Available: http://yaml.org/

[218] C. Evans. (2001, June 24). *MongoDB Regular Expression Capabilities*. Available: Available from: http://yaml.org/

[219] (2019, June 24). *SNORT*. Available: https://www.snort.org/

[220] D. Reisnger. (2019, June 24). *A.I. Expert Says Automation Could Replace 40% of Jobs in 15 Years*. Available: http://fortune.com/2019/01/10/automation-replace-jobs/

[221] Y. Freund, "An adaptive version of the boost by majority algorithm," *Machine learning,* vol. 43, pp. 293-318, 2001.

[222] L. Breiman, "Bagging predictors," *Machine learning,* vol. 24, pp. 123-140, 1996.

[223] R. Kohavi, "Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid," in *Proc. ACM 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, Portland, Oregon, 1996, pp. 202-207.

[224] O. Kramer, "K-nearest neighbors," in *Dimensionality reduction with unsupervised nearest neighbors*, ed: Springer, Berlin, Heidelberg, 2013, pp. 13-23.

[225] P. Peduzzi, J. Concato, E. Kemper, T. R. Holford, and A. R. Feinstein, "A simulation study of the number of events per variable in logistic regression analysis," *Journal of clinical epidemiology,* vol. 49, pp. 1373-1379, 1996.

[226] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric environment,* vol. 32, pp. 2627-2636, 1998.

[227] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval," in *Proc. Springer European Conf. on Machine Learning (ECML)*, Chemnitz, Germany, 1998, pp. 4-15.

[228] A. Liaw and M. Wiener, "Classification and regression by randomForest," *R news,* vol. 2, pp. 18-22, 2002.

[229] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. Springer 19th Int. Conf. on Computational Statistics (COMPSTAT)*, Paris France, 2010, pp. 177-186.

[230] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," *Journal of National Taiwan University*, vol. 1, pp. 1-16, 2016.

[231] L. V. Morales, A. F. Murillo, and S. J. Rueda, "Extending the floodlight controller," in *Proc. IEEE Int. Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, USA, 2015, pp. 126-133.

[232] A. Rajab "Burst Header Packet (BHP) Flooding Attack on Optical Burst Switching (OBS) Network Data Set," U. M. L. Repository, Ed., ed. University of South Carolina, 2017.

[233] (2018, June 24). *Wireshark Go Deep*. Available: https://www.wireshark.org/

[234] 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," *European Journal of Operational Research*, vol. 269, pp. 760 - 772, 2018.

[235] J. D. M. Rennie, L. Shih, J. Teevan, and D. R. Karger, " Tackling the poor assumptions of naive bayes text classifiers, " *in Proc.* AAAI 20th Int. Conf. on Machine Learning, Washington, DC, USA, 2003.

[236] (2019, July 30). Network layer mitigation techniques. Available: https://www.imperva.com/learn/application-security/ddos-mitigation-services/

## Appendix

CyberPulse can be downloaded from the link below, it provides a .sh script which can be run on a Ubuntu shell. This package will automatically create network and start capturing the traffic, however, the user need to run the floodlight controller first. Various parameters can be changed using the config.json file.

https://app.box.com/s/igb3fhmotbi7as1b7zcy130k7jhz4hv3

**List of acronyms used in this research.**

| Symbol | Description | Symbol | Description |
|---|---|---|---|
| LFA | Link Flooding Attack | DT | Decision Tree |
| SDN | Software Defined Network | RF | Random Forest |
| IP | Internet Protocol | KNN | K-nearest Neighbors |
| 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 |
| Wi-Fi | Wireless Fidelity | LTE | Long Term Evolution |
| IXP | Internet Exchange Point | SSID | Service Set Identifier |
| API | Application Programming Interface | OLSR | Optimized Link State Routing |
| ASes | Autonomous Systems | VN | Virtual Networks |
| FIB | Forwarding Information Base | SVM | Support Vector Machine |
| IDS | Intrusion Detection system | CoDef | Collaborative Defense |
| OF | OpenFlow | VPN | Virtual Private Network |
| ROC | Receiver Operative Curve | IPS | Intrusion Prevention System |
| DNS | Domain Name System | AUC | Area Under Curve |
| DHCP | Domain Host Configuration Protocol | RFE | Recursive Feature Elimination |
| SYN | Synchronization | HTTP | Hyper Text Transfer Protocol |
| RE | Regular Expression | DFA | Deterministic Finite Automata |
| NFA | (Non-deterministic Finite Automata | IoT | Internet of Things |
| DDoS | Distributed Denial of Service | PCRE | Perl Compatible Regular Expressions |
| IT | Information Technology | ANN | Artificial Neural Networks |
| IPS | Intrusion Protection System | DP | Dropped Packets |
| PDR | Packet Drop Rate | MLP | |
| AP | Arrived Packets | SLR | Simple Logistic Regression |
| NB | Naïve Bayes | pkl | Python Pickle File |
| FTP | File Transfer Protocol | SGD | Stochastic Gradient Descent |
| SVC | Support Vector Classification | DT | Decision Tree |
| LR | Logistic Regression | SVM | Support Vector Machines |
| RFE | Recursive Feature Elimination | D4J | Deep Learning4j |
| BHP | Burst Header Packet | CRF | Conditional Random Fields |
| MTD | Moving Target Defense | PCA | Principal Component Analysis |
| FFNN | Feed Forward Neural Networks | PCRE | Pearl Compatible Regular Expressions |
| CMIR | Cache Miss to Instructions Ratio | OpenDDL | Open Data Description Language |
| HOCON | Human-Optimized Configuration Object Notation | LTE | Long-Term Evolution |
| REST | Resourceful State Transfer | AS | Autonomous Systems |
| UCI | University of California Irvine | PLR | Packet Loss Rate |
| PDR | Packet Drop Rate | LBR | Lost Bandwidth Rate |
| UBR | Utilized Bandwidth Rate | SDWMN | Software Defined Wireless Mesh Networks |
| PRR | Packets Received Rate | GUI | Graphical User Interface |
| POD | Ping of Death | 5G | Fifth Generation |
| CLI | Command Line Interface | RTT | Round Trip Time |
| WSN | Wireless Sensor Networks | VN | Virtual Networks |
| IoT | Internet of Things | TCAM | Ternary Content Addressable Memory |
| DCI | Deep Packet Inspection | EM | Expectation Maximization |
| SVD | Singular Value Decomposition | L-BFGS | Limited Memory Broyden–Fletcher–Goldfarb–Shanno |
| BFGS | Broyden–Fletcher–Goldfarb–Shanno | RBF | Radial bias function |
| MAXENT | Maximum Entropy Classifier | PGMs | Probabilistic Graphical Models |
| CNN | Convolutional Neural Network | CART | Classification and Regression Tree |
| NTM | Neural Machine Translation | AI | Artificial Intelligence |

| | | | |
|---|---|---|---|
| BT | Boosting tree | DAG | Directed Acyclic Graph |
| Aml | Ambient Intelligence | OpenDDL | Open Data Description Language |
| HOCON | Human-Optimized Config Object Notation | GRE | Generic Routing Encapsulation |
| CLI | Command Line Interface | MAXENT | Maximum Entropy Classifier |
| DCI | Deep Content Inspection | SSID | Service Set Identifiers |
| NBI | Northbound Interface | C-DPI | Controller-Data Plane Interface |
| A-CPI | Application-Control Plane Interface | FA | Finite Automata |
| NFA | Non-Deterministic Finite Automata | NIDS | Network Intrusion Detection System |
| SLA | Service Level Agreement | SDWN | Software Defined Wireless Networks |
| SDMN | Software Defined Mobile Networks | SDLAN | Software Defined Local Area Networks |
| FIB | Forward Information Base | ICMP | Internet Control Message Protocol |
| IXP | Internet Exchange Point | IDS | Intrusion Detection System |
| UDP | User Datagram Protocol | EM | Expectation Maximization |
| SSID | Service Set Identifier | GRE | Generic Routing Encapsulation |
| SOC | Security Operation Center | RTT | Round Trip Time |
| WAN | Wireless Area Network | 5G | Fifth Generation |
| BGP | Border Gateway Protocol | GSOC | Global Security Operation Center |
| CMIR | Cache Misses to Instructions Ratio | PCRE | Perl Compatible REs |
| EM | Expectation Maximization | PC | Principal Components |
| PCA | Principal Component Analysis | TP | True Positive |
| FP | False Positive | FN | False Negative |
| TN | True Negative | CRF | Conditional Random Fields |
| BT | Boosting Tree | JVM | Java Virtual Machine |
| CV | Computer Vision | RFE | Recursive Feature Elimination |
| AUC | Area Under Curve | ROC | Receiver Operative Characteristic |
| UCI | University of California, Irvine | | |