AutoDrop: Automatic DDoS Detection and its Mitigation with combination of sFlow and OpenFlow

Author

Faisal Shahzad

Registration Number

2011-NUST-MS PhD- CSE(E)-10

Supervisor
Dr. Shoab A. Khan

Co-Supervisor
Dr. Muazzam A. Khan

DEPARTMENT OF COMPUTER ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

2016

AutoDrop: Automatic DDoS Detection and its Mitigation with combination of sFlow and OpenFlow

Author

Faisal Shahzad

Registration Number

2011-NUST-MS PhD- CSE(E)-10

A thesis submitted in partial fulfillment of the requirements for the degree of

MS Computer Software Engineering

Supervisor:

Dr. Shoab A. khan

Supervisor's Signature:_____

Co-Supervisor:

Dr. Muazzam A. khan

Co-Supervisor's Signature:_____

DEPARTMENT OF COMPUTER ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

2016

Declaration

I certify that this research work titled "AutoDrop: Automatic DDoS Detection and its Mitigation with combination of sFlow and OpenFlow" is my own work. I hereby declare that I have developed this thesis entirely on the basis of my personal efforts under the sincere guidance of my supervisor Dr. Shoaib Ahmad Khan and co-supervisor Dr. Muazzam Ali Khan. All of the sources used in this thesis have been cited and contents of this thesis have not been plagiarized. No portion of the work presented in this thesis has been submitted in support of any application for any other degree of qualification to this or any other university or institute of learning.

Signature of Student

_____

Copyright Statement

Acknowledgments

All praises to Allah, the most kind and loving and the creator of this universe, for giving me courage and leading me through. This dissertation would not be possible without the help of ALLAH Almighty and after Him so many people in so many ways. I honor all of them for helping me out in my research work.

I would like to express my gratitude and appreciation to my supervisor Dr. Shoab A. Khan and co-supervisor Dr. Muazzam A. Khan, for their guidance, constant support, encouragement and enduring patience. I gratefully acknowledge my committee members; Dr. Farooque Azam, Dr. Saad Rehman and Dr. Arslan Shaukat for their support and valuable guidance.

I would like to express my deep love and gratitude to my loving mother and supportive father. My deepest love for my loving brother, caring sisters for their never-fading love, care, understanding and encouragement.

ABSTRACT

World has emerged into global village due to connected internet. Everyone is connected with each other through internet which make everyone subject of being compromised. Many organizations' confidential and private data has been compromised and many online services are compromised due to cyber-attacks. Many researches and innovation has been made for making network secure but commercial routers limit them to deploy custom security algorithms in real network. Recently, researchers succeed to innovate a novel protocol OpenFlow in Software Defined Networks. OpenFlow separates control plane and data plane and provide space for researchers to experiment their innovations. Control plane is responsible for all forwarding logic, which is managed by Controller (external server) and router just need to forward data packets. Due to this separation, now researcher can easily implement their own data monitoring, encryption and forwarding algorithms. Taking advantage of this innovation we used Controller (control plane) to analyze real-time traffic, detect DDoS attack and mitigate attack.

We surveyed literature and highlight many solutions but most of them worked on attack detection irrespective care of computation and performance impact. They also does not focus on attack mitigation and left it for future work. Keeping all this in view we worked on one comprehensive solution which real-time monitors traffic, identify anomalies and mitigate attacking source without impacting on network performance.

We used sFlow-RT analytic engine for real time network traffic monitoring. Multiple triggers can be registered using its REST API based on different thresholds for different types of attacks such as Ping Flood, SYN and Ping of Death. Whenever traffic deviates specified thresholds it trigger an event, our mitigation application handle this event and manage network state using Kinetic Controller.

Key words: DDoS, SDN, OpenFlow, sFlow, Software Defined Network, Security, Denial of Service

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1. Introduction

Many network applications, now-a-day, are real time in nature. In a real time application, a good response time (efficient) and high latency rate of user's requests are expected. Meeting the user's expectations and needs of efficient response time is a major issue to be addressed in real time networks, where the traffic rate is also high, but it becomes really difficult to maintain a reasonable traffic flow in such applications when the unwanted software attacks are thrown to the system. These unwanted attacks make a network system slow or sometimes totally unavailable for its intended users. Our, research objective is to address this Distributed Denial of Service (DDoS) problem in Software Defined Network (SDN) architecture.

In this chapter, first, the problem of Distributed Denial of Service (DDoS) problem in Software Defined Network (SDN) architecture is stated briefly. Then the motivation behind his research and objective of proposed architecture are explained. At the end, research contributions and structure of dissertation is listed.

1.1. Software Defined Network

Software Defined Network (SDN) has emerged in last two decades since 1990. Researchers were eager to make network programmable. In start, from 1990 to 2000, many small functions/ scripts were developed to automate the network. From 2000 to 2007, researchers focused on separating control plane and data plane. In 2006, Martin Casado, PhD student at Standford University in Silicon Valley developed something called Ethan. Later on, Standford and University of California, Berkeley did a joint research and standardize protocol with the name of OpenFlow [1].

Using this protocol in SDN, switch can be configured with a controller (server) and incoming packets are not required to be processed by switch. It looks into lookup table, if any match exists in the lookup, switch forwards packets as per lookup match. If there is no

match for incoming packet, it sends packet to controller for processing. Controller processes the packet and decides whether packet should be forwarded or dropped. In this process, control plane and data plane are separated.

SDN architecture is dynamic, manageable, scalable, cost-effective and adaptable. It can have multiple controllers in a network and multiple switches can be configured with a single controller. This makes network control programmable because, it is decoupled from data plane. Using this approach, administrators can adjust network wide traffic policies dynamically to meet the changing needs of the network traffic. It also manages global view of network and presents it as a single logical switch.

1.1.1. OpenFlow

OpenFlow is an open standard that enables researchers to run experimental protocols in the campus networks. OpenFlow is used at commercial level and implemented by different vendors. It is embedded in different Ethernet switches, routers and wireless access points. It provides a research platform to run experiments without exposing the networks' internal details.

In a traditional router or switch, high level routing decisions that are control path, and packet forwarding task which is commonly known as control path occur on the same device. These two functions are performed by two different components in an OpenFlow switch. High level routing decisions are taken by a separate controller which is mostly a standard server while data path is still handled by switch as shown in Figure 1.1. The communication between OpenFlow switch and controller is based on OpenFlow protocol. OpenFlow uses different messages for communication like packet-received, modify-forwarding-table, send-packet-out, and get-stats.

OpenFlow facilitates the researchers to innovate routing and switching protocols in networks. Virtual machine mobility, high security networks and next generation IP based mobile networks are the applications which use OpenFlow networks commonly [2].

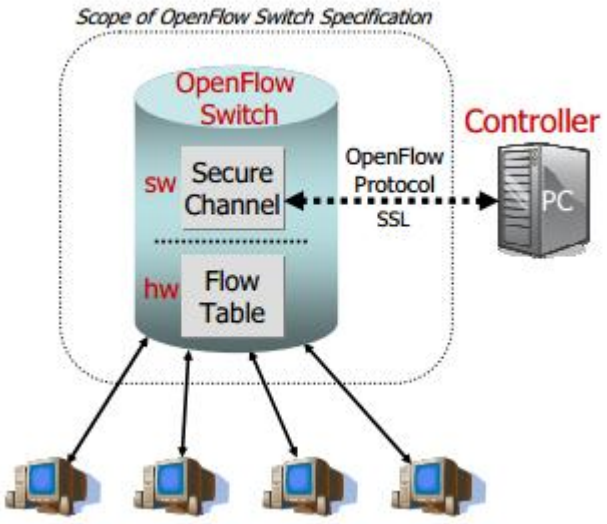


Figure 1.1. Flow Table is controlled by a remote controller via the Secure Channel [2]

1.1.2. OpenFlow Specifications

An OpenFlow switch consists of flow table and secure channel. Packet lookup and forwarding are performed by flow table while switch is managed by controller using OpenFlow protocol over a secure channel. Flow table hold three type of information; a set of flow entries known as header fields, activity counters and a set of actions that can be applied on matching packets. Flow table structure is shown in Table 1.1.

Header Fields	Counters	Actions

Table 1.1. A flow entry consists of header fields, counters and actions

a) Header Fields

Header fields of each incoming packets are checked in lookup table to find the matching entries. Each entry contains either a specific value or the string ANY which means that any value here is considered to be a matched value. If subnet masks on IP source or destination fields or both are specified on switch, then matches can be specified more accurately. Switch designers can implement the switch structure as per their convenience as far as correct functionality of switch is maintained. For example, different forward actions can be

specified while each indicating a different port. These different actions can be implemented using a single bitmask in the hardware forwarding table by the switch designers.

b) Counters

Counters are maintained per-table, per-flow, per-port and per queue. OpenFlow counters can be implemented in software and maintained by polling hardware counters with more limited ranges. A time is installed in the switch which refers the durations of the flow. Different explicitly specified errors like, CRC, overrun and frame errors are specified in Receive Errors field. In this document, a byte refers to 8-bit octets. Counters wrap around without any overflow indication.

c) Actions

Each entry in action attribute can have zero or more actions each indicating how switch handles matching packets. The packet is dropped in the case when forward actions are absent. The processing on the inserted flow entries must be specified according to the order of actions listed in action lists. Though, the order of packet output is not guaranteed within a port. That's why it is quite possible that two packets are sent to two different VLANs on a single port based upon the action list. These two packets can be re-ordered randomly, but the packet bodies must match the entities generated from the actions executed sequentially. A flow entry might be rejected by a switch if the switch cannot execute the actions specified in listed order. In such a case an unsupported flow error should be returned by the switch.

1.2. DDoS Attacks

Denial-of-service (DoS) or distributed denial-of-service (DDoS) attack is an attempt to degrade a system or a network performance in such a way that it becomes unavailable to its deliberate users. There are numerous methods being used for compromising resources and degrading performance of online available services. These attacks can be classified in two main categories:

1. Network Level Attacks
2. Application Level Attacks

1.2.1. Network Level Attacks

Network level attacks generate high traffic rate or bandwidth to make network busy for serving its desired job e.g. Ping Flood, TCP SYN Flood, Ping of Death etc. Once an attacker has got the access to a network, he can harm the system in any of the following ways:

- Attention of Information Systems staff is randomized in such a way that interruption cannot be seen immediately, mean while more attacks are thrown by the attackers when the distraction phase
- Terminating or altering the behavior of an application or a service in an abnormal way by sending invalid data to application or network service
- Flooding a machine or a network with fake traffic until a shutdown occurs because of traffic overload
- Network traffic blocking in such a way that the authorized users can not access the network resources

Packet rate and bandwidth rate are used to identify the types of attacks.

1.2.2. Application Level Attacks

An application layer attack tries to degrade the function of application servers by intentionally injecting faults in server's operating system or applications. As a result of application level attacks, the attacker gets the ability to bypass normal access control. The attacker takes advantage of the situation by gaining a control to a system, network or application and can perform following actions.

- Read, add, delete, or modify data or operating system
- Launch a virus program that uses network and system resources to copy viruses throughout network
- Start a sniffer program to investigate your network to get secret information about system that is eventually used to crash and corrupt the system and whole network

- Termination of data applications and operating systems abnormally
- To enable future unwanted attacks to system, security controls are disabled

These types of attacks are difficult to detect on network level as the deviation caused by these attacks is not too much visible on the network.

1.3. sFlow (Network Monitoring Tool)

Business critical applications reliance on network services is increasing day by day, due to this reliance a smallest change in system setup can impact network reliability and performance. Key business functions and cost of maintaining network services is directly influenced by degradation in network system performance and security.

By providing extraordinary visibility into network usage and active routes of high speed and complex networks, sFlow manages data to control network usage effectively, which provides competitive advantage.

Following are few applications of sFlow data:

- Detecting, diagnosing, and fixing network problems
- Real-time congestion management
- Usage accounting for billing and charge-back
- Understanding application mix (eg P2P, Web, DNS etc) and changes
- Identification of unauthorized network activities and tracing of denial-of-service attacks sources
- Route profiling and peering optimization
- Trending and capacity planning

sFlow is a sampling technology that meets the key requirements for a network traffic monitoring solution:

- sFlow is provided by a wide range of network equipment and software application vendors with interoperable implementations
- sFlow is a scalable technique for maintaining network traffic. Traffic data is measured, stored and analyzed by sFlow. Active routes and network wide view of usage is maintained by sFlow. This enables a huge amount of interfaces to be supervised and managed from a single place
- sFLow is a flexible and scalable mechanism to monitor high speed links without degrading the performance of internet routers and switches. Any additional network load is not added into system due to sFlow
- Flow is an inexpensive solution. Due to its low cost nature, it is preferred and used by a wide range of devices, from simple L2 workgroup switches to high-end core routers. Any additional memory or CPU utilization is not required by sFlow

1.4. Utilization of OpenFlow in DDoS Mitigation

OpenFlow interacts with two layers of OSI model; network and transport layers. First the properties of a packet are matched with rules and then the packet is forward to transport layer. As the OpenFlow cannot monitor the application level attacks so our research targets the utilization of OpenFlow for detecting network level attacks and their mitigation.

1.5. Problem Statement & Motivation

Distributed Denial of Service (DDoS) attacks is a well known mechanism to degrade the availability of network systems and make it difficult for the intended users of a system to get the services of a system. There are many different types of DDoS attacks stretching from high volume flooding on the system to the abuse of use level vulnerabilities. Although it is

quite easy to monitor the availability of a system services and network congestion but still it is a challenge to identify the accurate attackers in a DDoS.

Software Defined Networking (SDN) advances like OpenFlow gives more flexibility to control switching and routing mechanisms. The target of research conducted is to lessen the chances of DDoS attacks affecting the system. First we have investigated that how OpenFlow featured can be used to identify traffic spikes, identifying suspicious traffic and dropping unwanted traffic. We propose and implement a DDoS mitigation solution that uses these mechanisms and the custom algorithms that are executed in our OpenFlow controller.

1.6. Objective

In this research, we have targeted to find the automated solution for intrusion detection and its mitigation in real time using SDN. We explored different existing DDoS solution using SDN. During our study, we observed the following key points:

- None of them are providing complete solution from DDoS attack detection to its mitigation
- None of the solution is comprehensive enough that it gives a package to address the problem completely; from DDoS attack detection to its mitigation
- Most of the solutions degrade system performance
- Most of them limit their work to DDoS detection, leaving its mitigation on future work

Keeping these limitations in view, we studied different OpenFlow controllers and network monitoring tools and as a result we came out with a solution starting at DDoS attacks detection and ending at its attacks mitigation. Our research objective is, to provide an automated solution which detects DDoS attack in real time and immediately mitigates its source.

1.7. Research Contribution

In this research, we have proposed fully automated solution which is sufficient for effectively detecting DDoS attack and mitigating its source in real time using OpenFlow and sFlow.

This research identifies different existing solutions in our domain and critically reviews their capabilities and limitations. Our main contribution in this research is, to show how, we can utilize sFlow-RT analytical engine for DDoS detection and its triggers for generating events through which we can control network state using Kinetic controller. The key contributions are as follows:

- Proposed a lightweight efficient complete solution for unwanted attacks detection and mitigation
- Implemented proposed mechanism in Mininet
- Graphically demonstrated the working of proposed solution
- Showed the effectiveness of proposed solution through results simulation

1.8. Thesis Structure

This dissertation is organized as, Chapter 2 covers overview of SDN, different existing OpenFlow controllers and author controller selection, DDoS attack, network monitoring tool i.e. sFlow-RT and about Finite State Machine (FSM). Proposed architecture and its all components are explained in chapter 3. Chapter 4 demonstrates the execution of the system. Conclusion of work done and its possible extensions in future are stated in chapter 5. At the end there are references of all the research papers cited in this dissertation.

Chapter 2. LITERATURE REVIEW

In this chapter, a review of existing literature was performed to support the study undertaken in this dissertation. A general survey was conducted on past research efforts on network level security, software defined network, OpenFlow protocol, network traffic monitoring, network level application development frameworks and existing comprehensive solutions for intrusion detection.

2.1. Network Security

Yao et el., proposed Virtual source Address Validation Edge (VAVE), a method for securing network from spoofed IPs. They suggest mitigating unwanted attacks using OpenFlow devices rather than SAVI devices as shown in Figure 2.1 [3]. They pointed out that SAVI devices are good to detect spoofed IPs but each SAVI device is working independently without collaborating or using other device knowledge because they cannot communicate with each other which cause re-calculation on each device again and again.

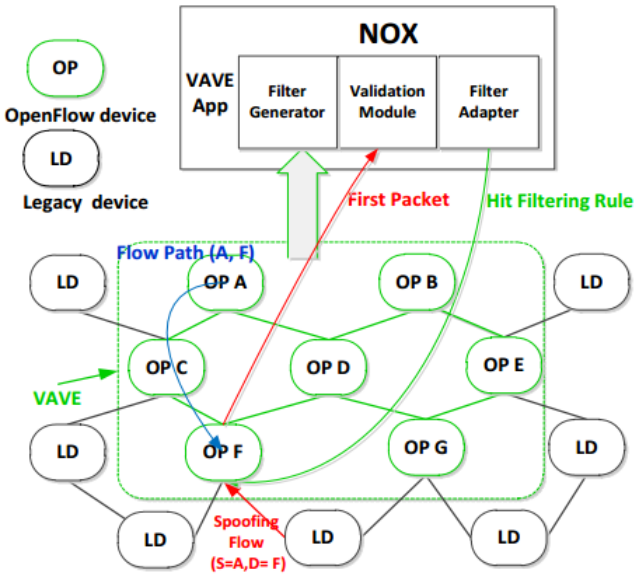


Figure 2.1. The Architecture of VAVE [3]

In their solution, VAVE, they used OpenFlow devices with central controller and formed a perimeter. When packets enters the perimeter, first OpenFlow device apply validation filters on packet using NOX controller, remaining all just used that information. The module uses "Packet In" to validate packet with white list addresses.

To reduce the processing load, they have installed all rule filters for detecting and dropping packets in advance. VAVE only detects IP spoofing attacks with pre-defined filtering rules; it cannot detect intrusion based attacks like Ping of Death, TCP SYN and Ping Flood.

Shin et al., proposed CloudWatcher, another commonly used network security solution. They proposed a simple scripting language to help network operators defining the network security policies. In this design, OpenFlow controller does not have any security module; instead they used other appliance for network security such as intrusion detection system. In this system, OpenFlow captures incoming network packets and forwards them to security appliances that inspect all of them shown in Figure 2.2 [4].

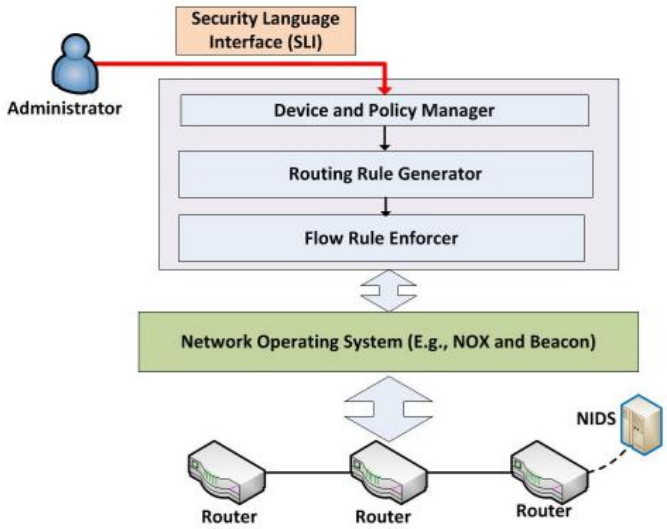


Figure 2.2. CloudWatcher Architecture [4]

This solution helps to enforce all incoming traffic to pass through configured security appliance. OpenFlow forwards each packet to security appliance for inspection and each packet is being inspected before forwarding. This impacts the network performance i.e. 1000 packets per second.

2.2. OpenFlow Custom Switch

Instead of OpenFlow controller, Kumar et al., proposed an OpenFlow switch for intrusion detection by adding two more tables for attacker IP (IDS IP) and signatures of malicious attacks.

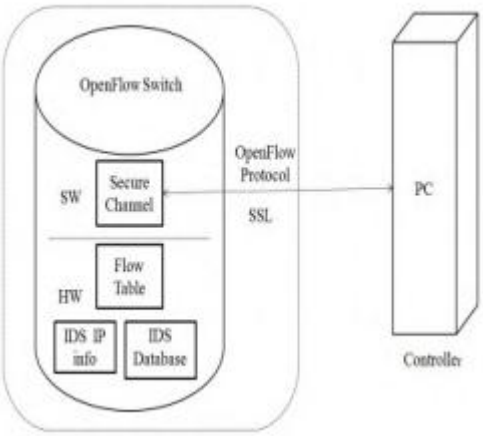


Figure 2.3. S. Kumar et. al., proposed OpenFlow switch with IDS detection [5]

First they look into IDS IP table, if packet IP is found there, then they simply drops the packet. If match not found then they look into IDS signatures table to check whether the packet is malicious or not. If packet is found to be malicious, they add it to IDS IP table for future use as shown in Figure 2.3 [5].

This solution greatly helps in

1) Real-time packet inspection and validation
2) Real-time attack mitigation
3) Secure and intelligent network availability

On the other hand, this solution also impacts the system in following ways:

1) Number of packets processed per second
2) Traffic flow per second

3) Cause bottle-neck in the network

4) Greatly impact on network performance

Fahimeh Alizadeh et al, researched on a hybrid NIPS/NIDS for terabit networks. They used BroCluster and SNORT with the combination of OpenFlow to identify and prevent DDoS attack. They used OpenFlow for load balancing and traffic reengineering. They did not utilize OpenFlow itself for identifying DDoS attacks instead they used SNORT which is freely available for intrusion detection [6].

2.3. Collaboration with ISP

Jun li et al proposed DrawBridge, based on the assumption that the customer's controller can communicate with ISP controller. This enables customers to subscribe for ISP's traffic engineering service. Customer expresses its traffic engineering policies to DrawBridge controller in ISP; the DrawBridge controller further pushes these policies to SDN switches deployed in ISP network to filter traffic or another DrawBridge controller in the ISP upstream as shown in Figure 2.4 [7].

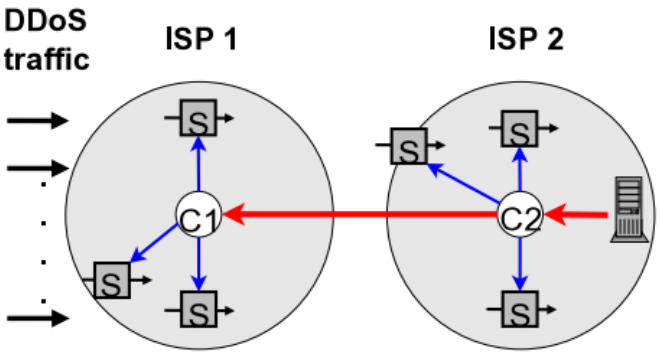


Figure 2.4. DrawBridge Proposed Solution [7]

They preferred to throttle traffic rather than blocking attacking node. They suggested to install rules on ISP hosted OpenFlow controller. They are throttling traffic using OpenFlow protocol instead of dropping useless traffic. This way the useless traffic is still coming in the network and network resources get busy in useless work.

Based on the same assumptions as DrawBridge, Rishikesh S. et al proposed Autonomic DDoS Mitigation using software defined network. They proposed that DDoS mitigation can be autonomous application as a service which can be consumed by customer network and multiple ISP networks. Both customer and ISP should have their own DDoS detection engine. They can alert each other about attacks and can use middle-box mitigation application as shown in Figure 2.5 [8].

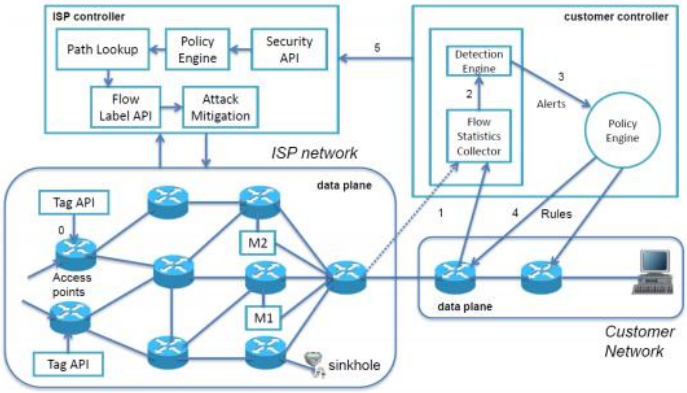


Figure 2.5. Customer-oriented Collaborative DDoS Mitigation Framework: spanning from one customer network to (multiple) ISP networks [8]

They proposed a novel change in the framework; which reduces the effort of developing mitigation application individually. But there will be also chances of one point failure for multiple infrastructures.

2.4. Network Monitoring

P. Phaal et al, demonstrates commercially available network monitoring tool sFlow; which is helpful for monitoring traffic in data network containing switches and routers. sFlow uses sampling techniques for continuous monitoring site-wide traffic for high speed switched and routed network. sFlow monitoring system consists of sFlow agent and a central data collector or sFlow analyzer. Flows can easily register with sFlow analyzer for identifying network anomalies [9].

S. Rehman et al, has proposed a flow monitoring tool OF@TIEN for network wide traffic visibility using sFlow monitoring tool.SDN flow monitoring application gets slice flow definitions from OpenFlow controller, loads them into sFlow-RT, fetches summary statistics

and feeds them to Graphite real-time charting tool. Our monitoring system also enables us to monitor GRE tunnels which are used to isolate traffic of tenant networks [10].

2.5. OpenFlow Applications

S. A. Mehdi et al, proposed traffic anomaly detection on SDN environment. They used multiple anomaly detection algorithms for validating their suitability in small office/home office environment. Author suggests that the decentralized control of distributed low-end network devices using OpenFlow can efficiently detect network anomalies and limit network security problems. Their experimental result shows efficiency of their intrusion detection algorithms with low network traffic ranging from 60 to 12,000 packets per second. Moreover, they left mitigation of detected network anomalies on their future work [11].

S. M. Mousavi, researched on attack detection using controller rather than network because OpenFlow controller is back bone of SDN architecture. If someone targets OpenFlow controller by spoofed IP packets, then controller resources will be consumed by spoofed IP, it may cause out of service and by this way SDN architecture can be collapsed easily. He proposed entropy based light weight solution in the controller by just adding two code functions. But he does not focus on network, if some attack all hosts in the network then this system will not be able to detect attack [12].

Rodrigo B. et al, proposed a light weight DDoS flooding attack detection using NOX/ OpenFlow. They implemented northbound application for OpenFlow and used Self Organizing Map (artificial intelligence) algorithm for dynamically identifying DDoS attacks as shown in Figure 2.6 [13]. This technique requires prior training of algorithm and knowledge base for making decision. This training cannot be provided in home or small office network.

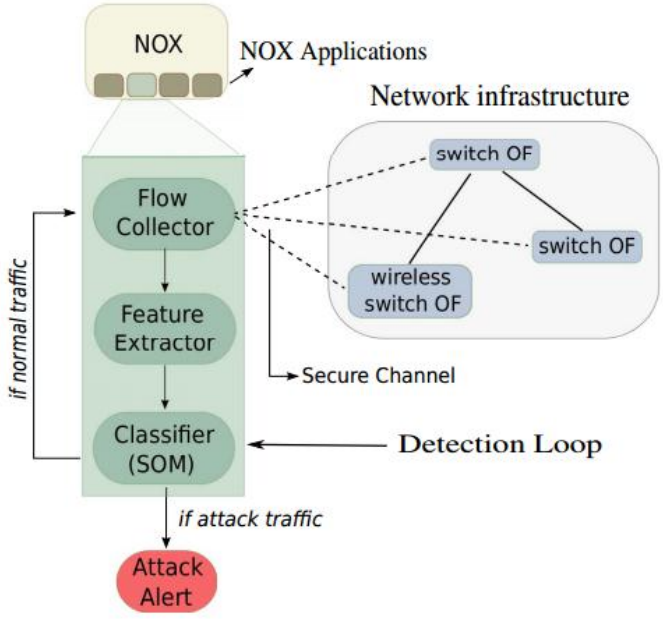


Figure 2.6. Rodrigo B. et. al., proposed Intrusion Detection Loop [13]

2.6. Development Framework

Proposals for secure SDN are not limited. S. Shin et al. focused on solution for developing and deploying complex OpenFlow security applications in much easier and rapid way. They presented FRESCO, a new application development framework, complete solution for designing, implementation and deployment of network security applications as shown in Figure 2.7. This framework helps developers in developing application from scratch to an enterprise level and also assists with some sample security applications. These applications are easily deployed as OpenFlow application [14].

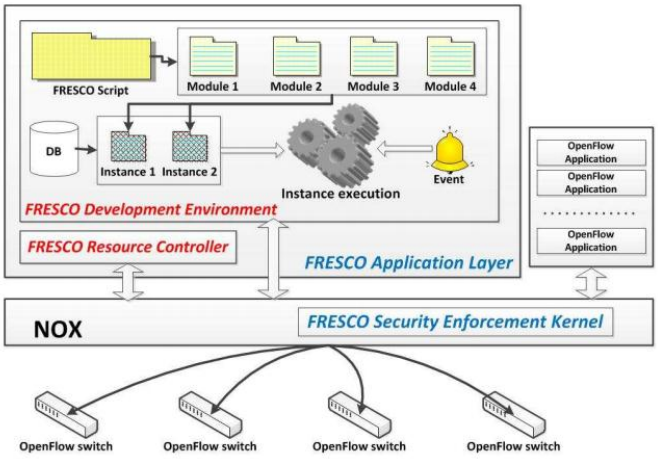


Figure 2.7. FRESCO Framework [14]

Looking at all above identified limitations, a novel architecture is experimented with the combination of OperFlow Northbound API, Kinetic (OpenFlow) Controller and sFlow (efficient network monitoring tool) with much higher network traffic up to 130,000 packets per second.

Chapter 3. Proposed Approach

In this chapter, we briefly explained our solution for DDoS attacks detection and their immediate mitigation. Fist, we have explained architecture overview including its all components then explained its execution process.

We propose real time traffic monitoring mechanism which monitors the traffic entering the customer network from ISP provider or internet exchange (IX) as shown in Figure 3.1. It monitors the statistics of traffic; passing through OpenFlow enabled switches, checks for anomalies and forwards only those packets which are from authentic users. If any malicious user found, it blocks the source and make it inaccessible for all.

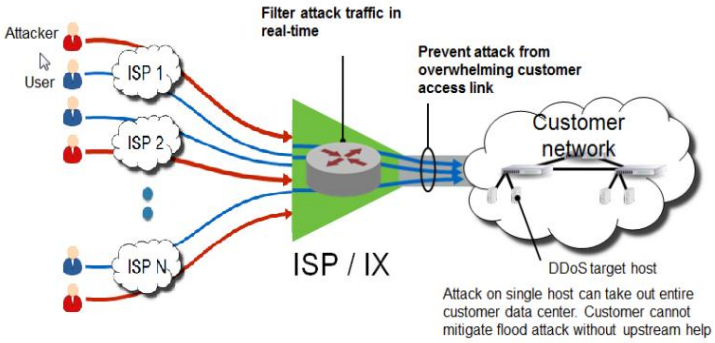


Figure 3.1. ISP Market View

Our proposed solution for DDoS attacks detection and their mitigation is based on *Separation of network monitoring, attack detection and attack mitigation.*

We propose a solution consisting on three modules. First module is responsible for monitoring traffic flows passing thru switch. It manages all statistics and generates samples for anomaly detection. Second module is responsible for managing network state and third module is responsible for registering events in first module. It generates events and operates network using second module.

- *Compatibility with any OpenFlow-enabled switches*

Our solution is based on SDN for attack detection and mitigation so any OpenFlow enabled switch can be used.

- *Efficient network monitoring for attack detection*

We require access to aggregate counters of each flow entry inside OpenFlow forwarding table for attack detection. OpenFlow only provides flow entries that correspond to aggregate flows, which does not meet the requirement of attack detection. S. A. Mehdi et. al., used OpenFlow for network monitoring and anomaly detection which has a high impact on the system performance [11]. OpenFlow results in congestion while the exhaustive message exchange between OpenFlow controller and OpenFlow-enabled device, when an attack is thrown. Instead of OpenFlow network monitoring mechanism, we have chosen an alternative tool sFlow for efficient network monitoring.

- *Real-time attack detection*

sFlow is multi-vendor sampling technology embedded within switches and routers. It is capable of monitoring application level traffic flows at wire speed continuously. It monitors all interfaces simultaneously. Open vSwitch are used to utilize both sFlow and OpenFlow protocol to communicate with sFlow Analytic Engine and OpenFlow controller. sFlow and OpenFlow controller use Open Northbound APIs to provide network statistics and control functionality to SDN applications such as Load Balancer, DDoS.

- *Real-time attack mitigation*

OpenFlow controller is responsible for control plane which forward or drop packets. Kinetic is an SDN network control system (event driven OpenFlow Controller) that allows operators to express dynamic network policies in a concise, intuitive way. Using Kinetic, network behavior can be changed dynamically based on different network events.

- *Scalability with varying traffic*

Our approach offload traffic monitoring to external component i.e. sFlow. sFlow uses packet sampling technique to minimize the flow related data.

3.1. Architecture Overview

Our system comprises of three major components as shown in Figure 3.2; Network Monitor, Network State Manager and DDoS Mitigation Application. Each one described as follows:

3.1.1. Network Monitor

This module is responsible for real-time network monitoring and managing statistics which are prerequisite of DDoS detection. Two different methods [8,9,10] were explored for network statistics, they used OpenFlow based networking monitoring which impact on switch robustness. We have chosen sFlow over OpenFlow network monitoring for its efficient and real time monitoring as well it is highly vendor recommended tool.

S. Rehmanet. al., described sFlow agents in network devices use random sampling according to the defined sampling rate and therefore, can be used to monitor high speed networks (Gbps speeds and higher) with quantifiable accuracy. sFlow agent is embedded in OpenFlow enabled physical and virtual network devices.

sFlow-RT is real-time analytics engine. We have used sFlow-RT as sFlow collector and analytics engine. It receives a continuous stream of sFlow data grams from network devices and converts them into actionable metrics that are accessible through REST APIs. REST APIs makes it easy for each application to configure flows, retrieve metrics, set thresholds, and receive notifications [10].

3.1.2. Network State Manager

This module is responsible for managing traffic flows over the network and mitigating the identified attack. We have chosen OpenFlow protocol for meeting this requirement. OpenFlow allows adding new entries into forwarding table and update existing ones for mitigating malicious attacks.

There are many OpenFlow controllers available but we have chosen Kinetic Controller due to its concise, intuitive way of expressing dynamic network policies. This is an event driven OpenFlow controller that allows to dynamically changing network behavior based on

various types of network events. Kinetic controller manages network state based on Finite State Machine (FSM) mechanism, which gives a concise logical understanding for making the policies [15].

3.1.3. DDoS Mitigation Application

This module has key responsibility for real-time DDoS detection and its real-time mitigation. This module communicates with both sFlow and Kinetic Controller using their REST APIs. It registers flows and threshold in sFlow-rt using its REST API. Multiple flows can be registered for different types of attacks with their corresponding thresholds. sFlow-rt generates events if traffic meets specified threshold. This module update network by passing Kinetic Controller about host and their status. Kinetic Controller drops all the packets coming from that specific host.

3.2. Attack Identification And Mitigation

In the previous section, we have briefly explained architectural overview of our proposed system and explained functionality of each module. In this section, execution of our application is explained.

Our system is middle-box between public internet (ISP, Internet Exchange etc.) and customers' private network. This system is responsible for monitoring traffic and identifying anomalies, entering into or being generated from inside customer's private network. Customer can manage and monitor its network statistics using Customer Portal which is being provided by sFlow technologies. Customer can access this portal over the internet using HTTPS secure protocol.

Network monitor, monitors all traffic passing through OpenFlow switch and maintain statistics asynchronously using real time sFlow. sFlow-RT analytics engine receives continuous stream of traffic and convert them into actionable metrics that are accessible through REST APIs. Triggers can be registered using these REST APIs [9].

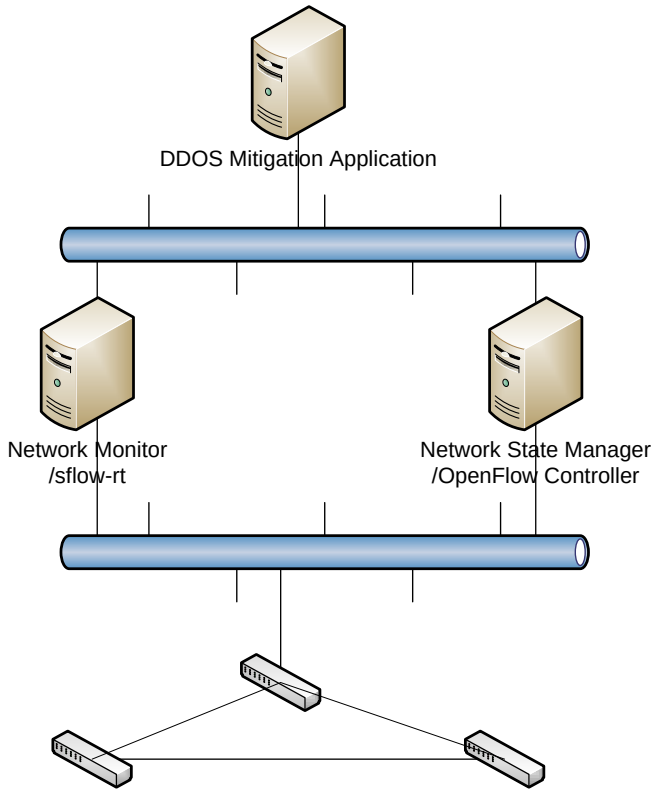


Figure 3.2. DDOS Mitigation Solution using sflow-rt and OpenFlow

DDoS mitigation application uses sFlow REST API to register thresholds on the basis of 1) number of packets being forwarded per second to identify Ping Flood attack, 2) size of packet for identifying Ping of Death attack, 3) number of connections being established to identify TCP SYN attacks.

Network Monitor, uses sampling technique with provided signatures and time interval to calculate aggregate statistics. These aggregate statistics are used to identify abnormalities in traffic flow. Events are raised for each threshold deviation. DDoS application catches these events and validates threshold value again to identify attack. In case of attack, DDoS application notify to Network State Manager.

Network State Manager is OpenFlow controller for managing network state. We are using Kinetic: Verifiable Dynamic Controller, which is developed for expressing network policies in a concise and intuitive way. This uses Finite State Machine (FSM)-based network policies which make very easy to compose policies. We used this to control our network state [15].

We have specified two network states and two policies i.e. Infected, Not Infected, identity and drop respectively. Infected or not infected specifies, whether source of coming request is infected or not and should we treat this packet as normal or simply drop because it's from infected source as shown in Figure 3.3.

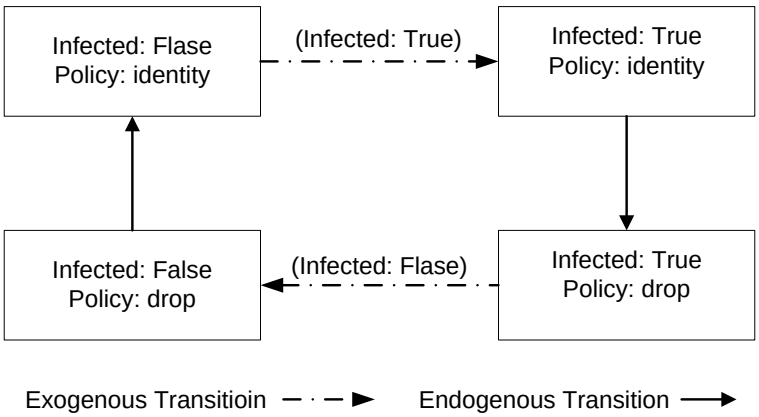


Figure 3.3. Finite State Machine (FSM) of Network

To demonstrate full execution of our proposed system, whole process can be briefly described in four steps as follows

1. **DDOS Attack**

Suppose normal traffic was passing thru switch as shown in start of Figure 14 and Figure15. Suddenly someone overloads customer port in customer's private network. You may assume any one attack of Ping Flood, Ping of Death or TCP SYN Flood attack.

2. **Attack Identification**

sFlow identifies large flows in the network as shown in Figure 14 and 15. By crossing specified thresholds, sFlow-RT detects attack by mapping large flows with in second as shown in Figures and it notifies DDoS Mitigation Application by generating events.

3. **Attack Mitigation**

Mitigation application receives attack event, it validate threshold value and applies customer network policies, select optimal control and push OpenFlow rule(s) to switch(es).

4. Add Rule in OpenFlow Forwarding Table

OpenFlow adds new rule(s) to switch (es) lookup table. By this way, OpenFlow switch(es), automatically apply these rules to all incoming packet and keep network protected.

Chapter 4. Implementation & Results

In this chapter, we have described about our experimental environment and output results with the comparison of results without presence of our application. We have elaborated our system with the execution of each attack and its mitigation separately.

4.1. Environment

It is not a good strategy (or it is not possible) to experiment a new approach, in a real network, because identifying the pros and cons of proposed technique separately, is not possible. Secondly, it may affect real legitimate traffic. So, with the invention of software defined network, researchers have developed a network simulation Linux based environment; Mininet, to experiment their ideas and new research methodologies.

Mininet is Linux based virtual machine build on Ubuntu 13.0.4. It uses Open vSwtich to implement realistic virtual network of hosts and switches. It is widely being used by SDN researchers because it can be easily configured on laptop to build realistic network topologies and experiment new methodologies with OpenFlow controller and SDN applications.

Many OpenFlow controllers have been proposed and being commercially used. We have chosen Kinetic controller due to its concise and intuitive way of writing network policies and managing network state using Finite State Machine (FSM). . In FSM, moving from one state to another state is quite easy. To write the policies, we are using Python language.

To collect real time network statistics, we have chosen sFlow-RT analytic engine. It monitors network traffic on real time and manages statistics. sFlow can be easily configured in mininet virtual machine with OpenFlow controller. We have chosen thresholds of different DDoS attack on the basis of mininet capacity, these thresholds can be modified in real environment and entropy based algorithms can be used for dynamic flexible thresholds.

sFlow supports JavaScript for registering flows and triggering events, so we have implemented our mitigation application in JavaScript. Mitigation application handles sFlow generated application and cross verifies thresholds value and manages network state using Kinetic controller. We are using node .js for execution of mitigation application because it is implemented in JavaScript.

4.2. Implementation

In this section, we have demonstrated execution of our proposed solution for each type of attack.

4.2.1. How to Run Application

To run application:- First in the ssh terminal, following command is executed to generate network topology in Mininet virtual network

sudo mn --controller=remote --topo=single,3 --mac --arp

In the second terminal run, the following command is exectuted to make sflow ready. Then the sflow monitoring tool is started.

sudo ovs-vsctl -- --id=@sflow create sflow agent=eth0 target=\"127.0.0.1:6343\" sampling=2 polling=20 -- -- set bridge s1 sflow=@sflow

cd sflow-rt

./start.sh

In the 3rd terminal run Kinetic Controller for managing network state

cd ~/pyretic

pyretic.py pyretic.kinetic.apps.ids

In the 4th terminal run DDoS mitigation application:

cd sflow-rt

nodejs extras/ddosmitigation.js

In the 5th terminal authenticate all hosts in topology to communicate through OpenFlow switch

cd ~/pyretic/pyretic/pyresonance

python json_sender.py -n authenticated -l True --flow="{srcip=10.0.0.1}" -a 127.0.0.1 -p 50001

python json_sender.py -n authenticated -l True --flow="{srcip=10.0.0.2}" -a 127.0.0.1 -p 50001

python json_sender.py -n authenticated -l True --flow="{srcip=10.0.0.3}" -a 127.0.0.1 -p 50001

python json_sender.py -n infected -l False --flow='{srcip=10.0.0.1}' -a 127.0.0.1 -p 50001

python json_sender.py -n infected -l False --flow='{srcip=10.0.0.2}' -a 127.0.0.1 -p 50001

python json_sender.py -n infected -l False --flow='{srcip=10.0.0.3}' -a 127.0.0.1 -p 50001

Following command is used to generate ping of death attack by sending 100,00,000 packets of size 65,500 bytes with 10 microsecond delay.

ping 10.0.0.2 -l 65500 -n 10000000 -w 0.00001

Following command is used to generate Ping Flood attack by sending 10,000 packets per second to provided IP

ping -i 0.0001 10.0.0.2

Following command is used to generate TCP SYN Flood attack by establishing 2000 connections at a time

sudo nping --tcp -p 80 --flags syn --rate 2000 --count 200000 10.0.0.3

sFlow monitoring module can be access with following link to visually analyze traffic statistics.

In the browser:

http://127.0.0.1:8008/metric/ALL/ddos/html

4.2.2. Ping Flood Attack

A ping flood is a denial-of-service attack where the victim is overwhelmed by the attacker ICMP Echo Request (ping) packets. ICMP packets are sent at high speed without waiting for the replies. Most implementations of ping require the user to be privileged in order to specify the flood option. If the network bandwidth of an attacker is higher than that of victim, this technique is most successful. An ICMP Echo Reply packet is expected by the victim and this consumes both the outgoing and incoming bandwidths. If the victim system is slow, it is possible to consume enough of its CPU cycles for a user to notice a significant slowdown.

In this section, we are using Ping Flood attack to elaborate execution of our system. To setup environment, first we built a virtual network topology; having one switch with three hosts having remote controller. Following Mininet command establishes this topology in virtual network

sudo mn --controller=remote --topo=single,3 --mac --arp

We may get a separate terminal for each host with following command

mininet>xterm host1

Let's flood the virtual network with Ping Flood (100,000 packets per second) using following mininet command.

ping -i 0.00001 10.0.0.2

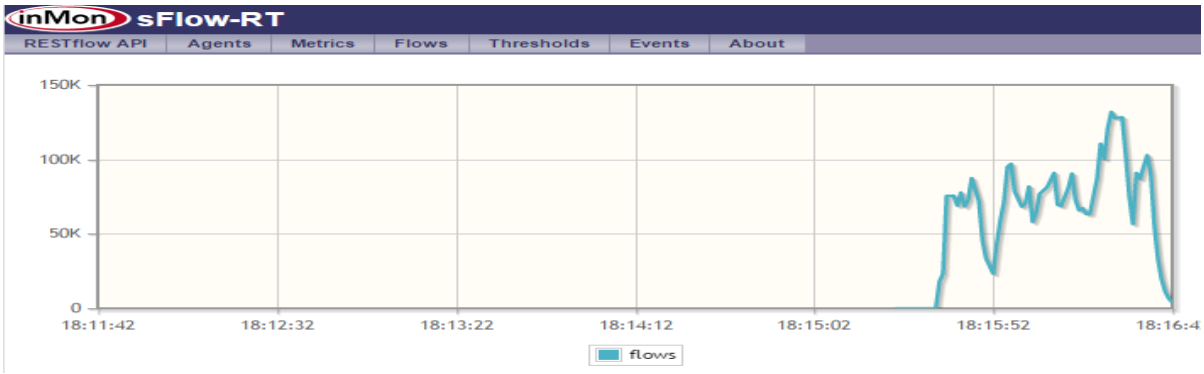


Figure 4.1. Network under Ping Flood attack without presence of our application

The sFlow-RT Figure 12 shows that ping flood attack generates around 90,000 packets per second traffic rate. Now stop the attack and configure sFlow for network monitoring. Following Mininet command configures sflow in Mininet environment

sudo ovs-vsctl -- --id=@sflow create sflow agent=eth0 target=\"127.0.0.1:6343\" sampling=2 polling=20 -- -- set bridge s1 sflow=@sflow

Following command starts sFlow

./start.sh

We have to configure Kinetic Controller for controlling network state

pyretic.py pyretic.kinetic.apps.ids

Further, run our mitigation application using nodejs third party tool

nodejs extras/ddosmitigation.js

Now start the Ping Flood attack again using same command

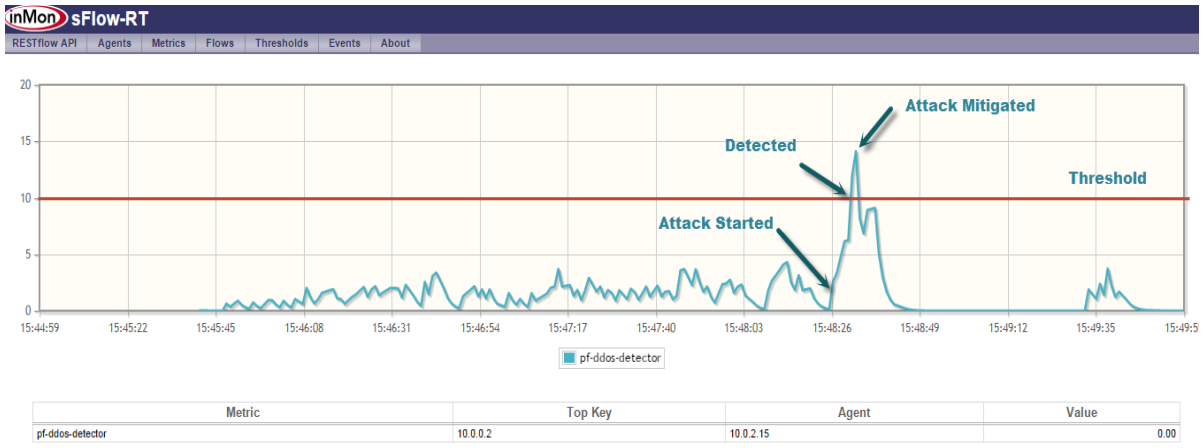


Figure 4.2. Attack Detection and Mitigation in presence of our application

sFlow-RT quickly detected ping flood attack and notifies the mitigation application. The mitigation application cross verify attack with specified threshold and construct following message to send it to Kinetic controller.

../pyretic/pyretic/kinetic/json_sender.py -n infected -l false --flow='{srcip=10.0.0.2}' -a 127.0.0.1 -p 50001

Kinetic controller pushes rule to Open vSwitch using OpenFlow which instantly starts dropping packets. Figure 13 shows, our system quickly detects when traffic exceeds specified threshold and immediately mitigates attack in the tenth part of second rather than reaching a peak of 80,000 packets per second. Attack is limited to a peak of 13 packets per second. We choose 10 packets per second threshold for demonstration purpose. This can be changed as per network traffic flow.

4.2.3. Ping of Death Attack

A type of attack in which an attacker attempts to destabilize, freeze or crash the target computer or service by sending oversize packets using a simple ping command. Traditionally, most computer systems cannot handle the ping packet larger than the maximum IPv4 packet size of 65535bytes. Larger packets can cause a system crash down. [16]

In this section, we are using Ping of Death attack to elaborate execution of our system. To setup environment, first we had built a virtual network topology; having one switch with three hosts having remote controller. Following Mininet command establishes this topology in virtual network

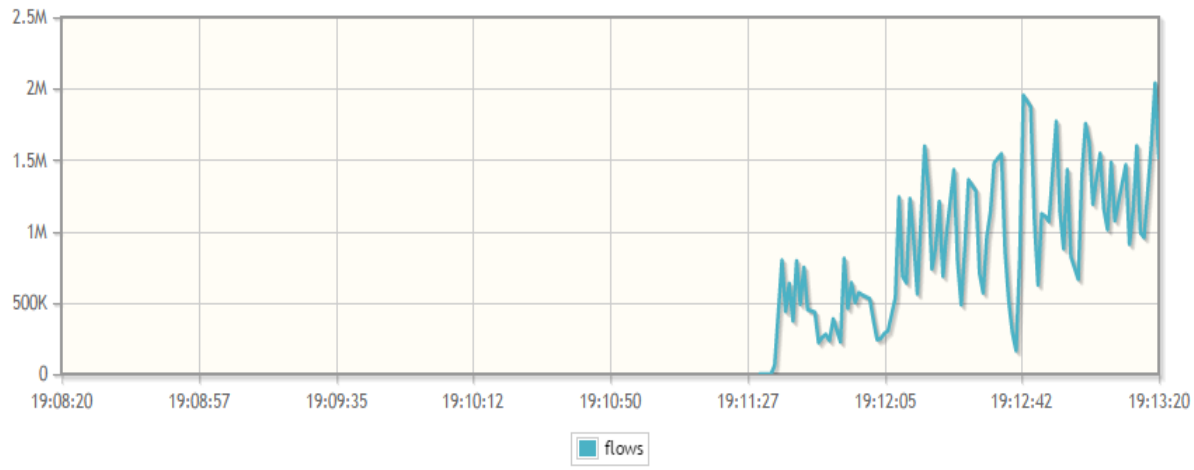
sudo mn --controller=remote --topo=single,3 --mac --arp

We may get a separate terminal for each host with following command

mininet>xterm host1

Let's flood the virtual network with Ping of Death (sending packet of size 65,500 bytes in every 10 micro seconds) using following mininet command.

ping 10.0.0.2 -l 65500 -n 10000000 -w 0.00001



Metric	Top Key	Agent	Value
flows	10.0.0.1,10.0.0.2,eth.ip.icmp	10.0.2.15	1.51M

Figure 4.3. Network under Ping of Death attack without presence of our application

The sFlow-RT Figure 4.3. Network under Ping of Death attack without presence of our applicationshows that ping of death attack sends around 1.5 MB packet at a moment. Now stop the attack and configure sFlow for network monitoring. Following Mininet command configures sflow in Mininet environment

sudo ovs-vsctl -- --id=@sflow create sflow agent=eth0 target=\"127.0.0.1:6343\" sampling=2 polling=20 -- -- set bridge s1 sflow=@sflow

Following command starts sFlow

./start.sh

We have to configure Kinetic Controller for controlling network state

pyretic.py pyretic.kinetic.apps.ids

Further, run our mitigation application using nodejs third party tool

nodejs extras/ddosmitigation.js

Now start the Ping of Death attack again using same command

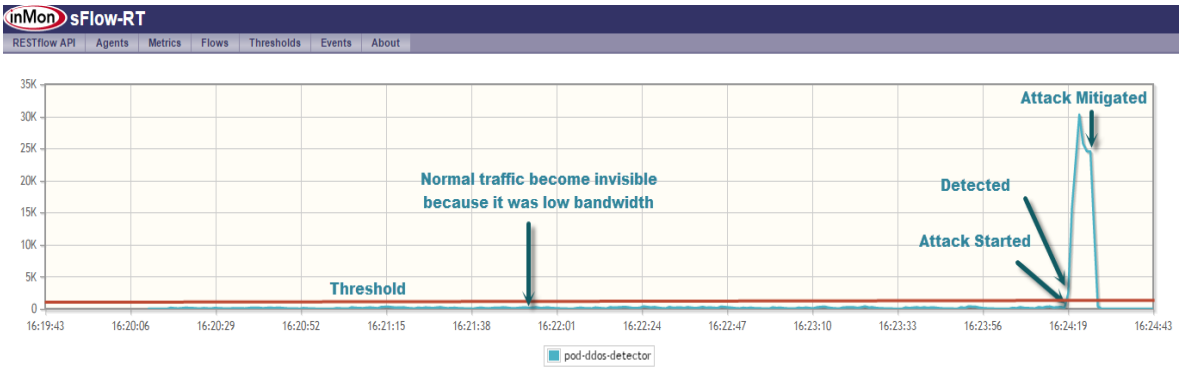


Figure 4.4. Ping of Death Attack Detection and Mitigation

sFlow-RT quickly detected ping of death attack and notifies the mitigation application. The mitigation application cross verify attack with specified threshold and construct following message to send it to Kinetic controller.

../pyretic/pyretic/kinetic/json_sender.py -n infected -l false --flow='{srcip=10.0.0.2}' -a 127.0.0.1 -p 50001

Kinetic controller pushes rule to Open vSwitch using OpenFlow which instantly starts dropping packets. Figure 4.4 shows, our system quickly detects when traffic exceeds

specified threshold and immediately mitigates attack in the tenth part of second rather than reaching a peak of 1.5 MB sized packets per second. Attack is limited to a peak of 25k bytes sized packets per second. We choose 25,000 bytes per second threshold for demonstration purpose. This can be changed as per network traffic flow.

4.2.4. TCP SYN Attack

TCP SYN flood (a.k.a. Synflood) is a type of Distributed Denial of Service (DDoS) attack. It exploits the TCP three-way handshake to consume network resources on the victim system and turn it into an unresponsive system. Using SYN flood DDoS, the attacker sends TCP connection requests on a speed faster than the processing speed of targeted machine. As a result, the victim machine becomes saturated.

When a normal TCP "three-way handshake" is established between client and server, the exchange looks like this:

1. Client requests connection by sending SYN (synchronize) message to the server.
2. Server acknowledges by sending SYN-ACK (synchronize-acknowledge) message back to the client.
3. Client responds with an ACK (acknowledge) message, and the connection is established.

In a SYN flood attack, repeated SYN packets are sent to the targeted server on each port, mostly using a fake IP address. The server receives multiple attacks, without knowing the fake requests. The server establishes the connection with fake requests, by considering them a legitimate request. It responds to each attempt with a SYN-ACK packet from each open port.

Let's elaborate the execution of our system by demonstrating TCP SYN flood attack. To setup environment, first we had built a virtual network topology; having one switch with three hosts having remote controller. Following Mininet command establishes this topology in virtual network

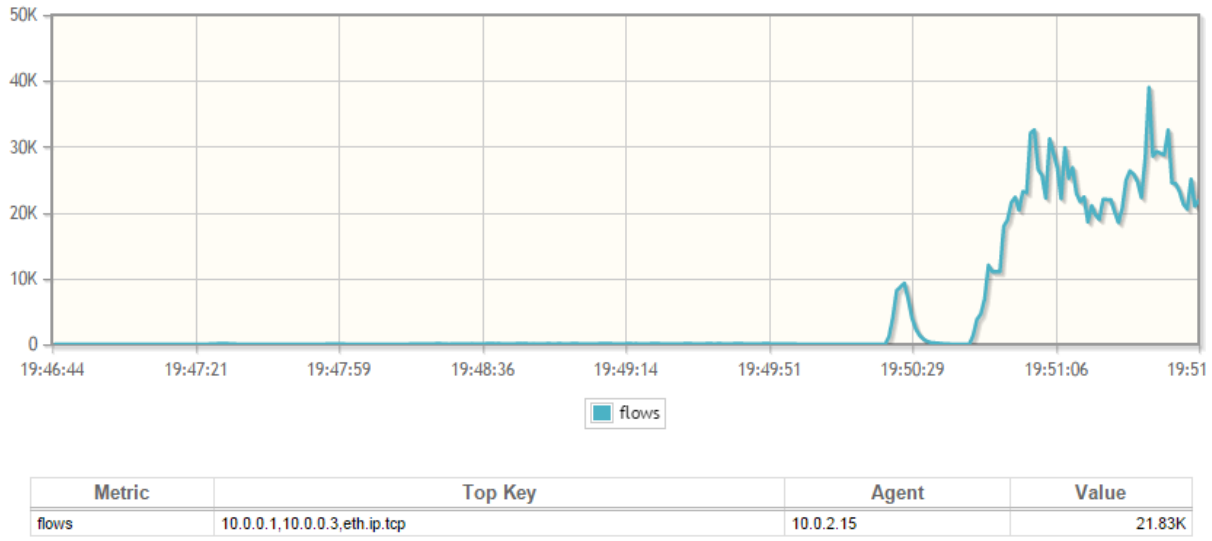
sudo mn --controller=remote --topo=single,3 --mac --arp

We may get a separate terminal for each host with following command

mininet>xterm host1

Let's flood the virtual network with TCP SYN attack (sending 2000 connections requests 200,000 times to host 3) using following mininet command.

sudo nping --tcp -p 80 --flags syn --rate 2000 --count 200000 10.0.0.3



Metric	Top Key	Agent	Value
flows	10.0.0.1,10.0.0.3,eth.ip.tcp	10.0.2.15	21.83K

Figure 4.5. Network under TCP SYN attack without presence of our application

The sFlow-RT Figure 4.5 shows that TCP SYN attack sends around 25K to 30K at a moment. Now stop the attack and configure sFlow for network monitoring. Following Mininet command configures sflow in Mininet environment

sudo ovs-vsctl -- --id=@sflow create sflow agent=eth0 target=\"127.0.0.1:6343\" sampling=2 polling=20 -- -- set bridge s1 sflow=@sflow

Following command starts sFlow

./start.sh

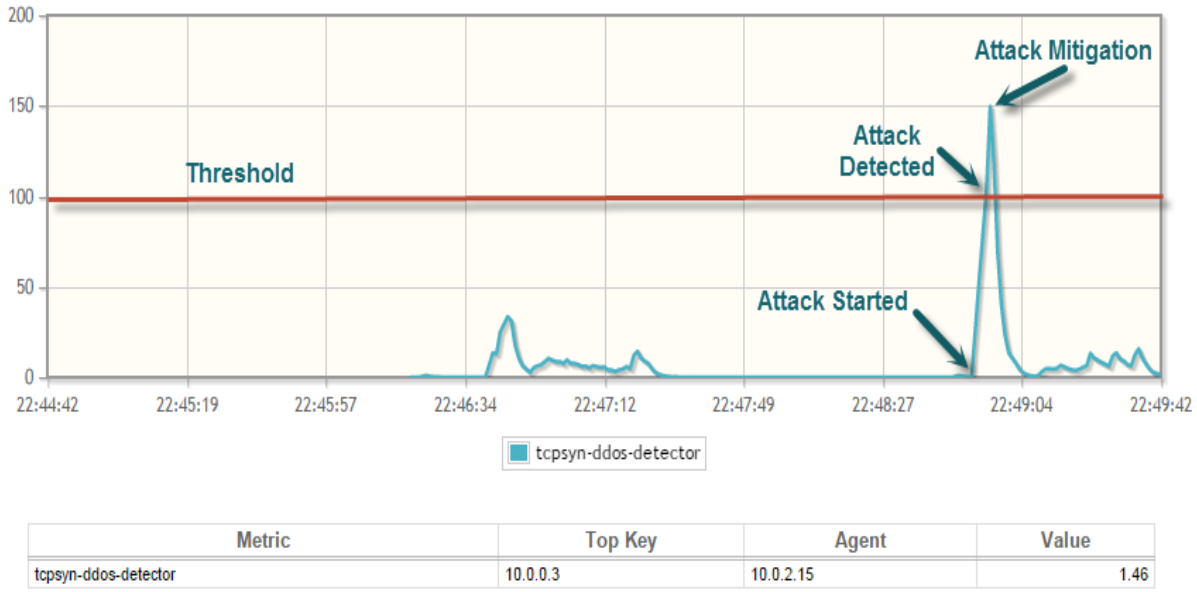
We have to configure Kinetic Controller for controlling network state

pyretic.py pyretic.kinetic.apps.ids

Further, run our mitigation application using nodejs third party tool

nodejs extras/ddosmitigation.js

Now start the TCP SYN attack again using same command



Metric	Top Key	Agent	Value
tcpsyn-ddos-detector	10.0.0.3	10.0.2.15	1.46

Figure 4.6. TCP SYN Attack Detection and Mitigation

sFlow-RT quickly detected TCP SYN attack and notifies the mitigation application. The mitigation application cross verify attack with specified threshold and construct following message to send it to Kinetic controller.

../pyretic/pyretic/kinetic/json_sender.py -n infected -l false --flow='{srcip=10.0.0.2}' -a 127.0.0.1 -p 50001

Kinetic controller pushes rule to Open vSwitch using OpenFlow which instantly starts dropping packets. Figure 4.6. TCP SYN Attack Detection and Mitigation shows, our system quickly detects when traffic exceeds specified threshold and immediately mitigates attack

in the tenth part of second rather than reaching a peak of 25K connections per second. Attack is limited to a peak of 100 connections per second. We choose 100 connections per second threshold for demonstration purpose. This can be changed as per network traffic flow.

4.3. Results

Attack detection time is inversely proportional to attack flow size (i.e. number of malicious packets being attacked per second) and directly proportional to threshold. Larger the number of malicious packets, shorter the detection time. For a large threshold value, duration of detection time is prolonged with a high attack flow size. Attack mitigation is independent of threshold and attack size. Its responsibility is to block attacking source within a second and it is blocking detected attack in tenth part of second using Finite State Machine technique. The Table 4.1 summarizes detection and mitigation time vs. flow size of our experiment:

Flow Size (Packets Per second)	**Threshold**	**Detection and Mitigation Time** (seconds)
10	100	100-120
100	100	85-90
150	100	23-25
200	100	18-20
10000	1000	1.819-2.017
100000	1000	0.753 - 1.149

Table 4.1. Flow Detection and Mitigation Time

The detection times shown in Table 4.1 with different sampling values are shown in Table 4.2.

Link Speed	Large Flow	Sampling Rate	Polling Interval
10 Mbit/s	>= 1 Mbit/s	1-in-10	20 seconds
100 Mbit/s	>= 10 Mbit/s	1-in-100	20 seconds
1 Gbit/s	>= 100 Mbit/s	1-in-1,000	20 seconds
10 Gbit/s	>= 1 Gbit/s	1-in-10,000	20 seconds
40 Gbit/s	>= 4 Gbit/s	1-in-40,000	20 seconds
100 Gbit/s	>= 10 Gbit/s	1-in-100,000	20 seconds

Table 4.2. sFlow Sampling Statistics

These sampling rates allow a central controller to monitor very large scale switch fabrics. In addition, we have applied multiple control functions in parallel based, on the sFlow data feed, to detect multiple types of DDoS attacks such as Ping Flood, SYN and Ping of Death.

4.4. Discussion And Comparison

There are various kinds of DDoS attacks; Ping Flood, Ping of Death and SYN Flood are most famous attacks in recent history. Many researchers have worked on different ways to identify DDoS attacks but most of them keep their focus on just attack detection rather than mitigating the attack source as well. Mitigation of attack source is also as important as its identification. Mostly they have focused on the attack detection without Worrying about the performance of the network. In VAVE, authors suggest to use OpenFlow vSwitch rather than using old SAVI switch because OpenFlow switches can be managed and controlled

centrally . They can share their rules with each other [3]. CloudWatcher proposed a scripting language for writing policies to pass all traffic from security appliances. DoS detection and mitigation will be responsibility of security appliance [4]. S. Kumar et. al. added two new lookup tables in switch for real time traffic monitoring and filtering [5]. But this can have a negative impact on the performance, as each packet is processed and then forwarded or dropped. DrawBridge proposed also to install rules on ISP OpenFlow controller rather than just installing on local OpenFlow controller [7].

Supporting DrawBridge solution, Rishikesh S. et al, proposed autonomic DDoS mitigation service which is the middle-box between customer network and multiple ISPs networks. Both customer and ISP have to subscribe central mitigation application for using this service. This is novel idea but relying on external mitigation application can be one point failure for multiple networks [8].

Rodrigo B. et al, proposed a novel solution for identifying network anomalies using self-organizing map but didn't focus on attack mitigation [13]. FRESCO provided a full development framework for developing network security applications which can be easily deployed over OpenFlow enabled network [14]. S. A. Mehdi et. al. also proposed a solution using different algorithms using OpenFlow. They monitor and process each packet for identifying whether it is malicious or not ? This approach processes 600 packets per second [11].

Controller has become back bone of software defined networks. If someone target controller and make it compromised then SDN architecture will be crashed. S. M. Mousavi researched and provided solution for securing controller by just adding two code functions [12]. Hetarget on securing controller only, rather than securing the whole network. If someone attacks whole network then controller will not be able to detect the attack effectively as shown in Table 4.3.

Considering performance as our key goal, with DDoS detection and its real time mitigation, we proposed a novel complete solution for DDoS detection and mitigation. Out System monitors network asynchronously and gather all traffic statistics using sFlow-RT. Our

system identifies three most famous DDoS attacks; Ping Flood, Ping of Death and SYN Flood on real time with performance varying from 80,000-130,000 packets per second.

	Impact on Performance	Ping Flood Detection	Ping of Death Detection	SYN Flood Detection	Spoofed IPs Detection	Attack Mitigation
VAVE[3]	-	×	×	×	✓	×
CloudWatcher[4]	✓	-	-	-	-	×
S. Kumar et. al extended OpenFlow Switch[5]	✓	✓	✓	×	×	✓
DrawBridge[7]	-	-	-	-	×	✓
Autonomic DDoS Mitigation [8]	-	✓	✓	✓		✓
Light weight DDoS flooding attack detection [13]	×	✓	✓	✓	×	×
FRESCO Framework[14]	-	-	-	-	-	-
S. A. Mehdi et. al. Anomaly Detection Module[11]	✓	✓	✓	×	×	×
Early Detection of DDoS[12]	✓	×	×	×	✓	×
DDoS Mitigation Application (Our System)	×	✓	✓	✓	×	✓

Table 4.3. Comparison with Literature

Chapter 5. Conclusion And Future Work

The whole research work is summarized in this chapter which includes the conclusion of the work done and the future directions for the extension and betterment of the system.

5.1. Conclusion

In this paper, we have evaluated Software Defined Network (SDN) for mitigating a great network threat by DDoS attacks using OpenFlow protocol. Our study demonstrated that entire network monitoring, on periodic basis including tenth of thousands of flows, does not scale for high traffic environment. Furthermore, using this technique a small medium flood attack may cause denial of service.

We proposed a novel solution which: (a) reduces data gathering overhead by using sampling technique implemented thru sFlow protocol, (b) detects anomalies using sFlow-rt analytics engine events, handled in most efficient JavaScript language (c) mitigates anomalies using OpenFlow protocol. We have offloaded OpenFlow for network monitoring with sFlow-rt it have/leaves impact on network traffic speed. Our system performance is not only comparable with that of OpenFlow technique for low traffic rate but also reliable for high traffic networks as well. Our Proposed and implemented system handles real time traffic ten times more efficiently than the existing techniques.

In this design, modules are loosely coupled and independent in their responsibility. Due to this, researchers can alter these modules for experimentation process using different algorithms and tools.

5.2. Future Work

OpenFlow is no longer merely of academic interest, as it is increasingly being deployed in real-world systems [17]. It has often been the case that technical standards have been focused on functionality at the expense of security. In the future, security should be

considered from the very first draft, not added on shortly before release or offered as an extension.

5.2.1. Entropy Based Threshold

In our research, we are using fixed approximate threshold for experimental basis. Threshold can be calculated on entropy based analysis, on runtime. Entropy is an important concept of information theory. It is a measure of randomness associated with a random variable or uncertainty. In our scenario, data coming over the network uses the concept of entropy, if data is more random it contains more entropy. The value of sample entropy lies in range [0, logn]. The rate of entropy is lesser when the class distribution is pure i.e.it belongs to one class. The rate of entropy is larger when the class distribution is impure i.e. class distribution belongs to many class. Changes in randomness are detected by comparing the rate of entropy of some sample of packet header fields to that of another sample of packet header fields [18].

5.2.2. Empirical Testing

This thesis makes use of Mininet 2.0 as a test environment for experimentation with OpenFlow. Although the primary intent is to detect and mitigate DDoS attack, it can also be used to model a wide range of other scenarios, as OpenFlow-managed networks can encounter scalability and performance issues in the absence of any malicious traffic. The proposed mitigation method has yet to be tested on production systems. It would be particularly desirable to have a test suite that could be used to validate switch and/or controller design, as a form of general stress test or as a specific unit test for particular vulnerabilities; the software developed could be a basis for this. Testing switch and controller design has also been the topic of other works [19]. The use of physical hardware would allow the impact of such attacks on real-world systems to be determined; there are significant differences in the behavior of a simulated network environment with little or no concurrent traffic and a real network environment with substantial non-malicious traffic and other performance constraints. Furthermore, the increasing use of virtual networks in cloud

systems makes testing attack scenarios on such systems vital - it is known that virtualization environments allow for novel attack types [20].

References

[1] Nick Feamster, Jennifer Rexford, Ellen Zegura, "The Road to SDN: An Intellectual History of Programmable Networks", ACM SIGCOMM Computer Communication Review 44.2, pp. 87-98, 2014.

[2] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, Jonathan Turner, OpenFlow: enabling innovation in campus networks, ACM SIGCOMM ComputerCommunication Review, v.38 n.2, pp. 42-47, 2008.

[3] Guang Yao, Jun Bi and Peiyao Xiao, "Source Address Validation Solution with OpenFlow/NOX Architecture", Proc. International Conference on Network Protocol (ICNP), pp.7-12, 2011.

[4] Seungwon Shin, GuofeiGu, "CloudWatcher: Network Security Monitoring Using OpenFlow in Dynamic Cloud Networks", Proc. International Conference on Network Protocol (ICNP), pp. 1-6, 2012.

[5]Suresh Kumar, Tarun Kumar, Ganesh Singh, Maninder Singh Nehra, Open Flow Switch with Intrusion Detection System, International Journal of Scientific ResearchEngineering & Technology (IJSRET), v 1, pp. 001-004, 2012.

[6] Fahimeh Alizadeh, Rawi Ramdhan, A Hybrid NIPS/NIDS for terabit networks, research report, delaat, 2013 (http://www.delaat.net/rp/2012-2013/p03/report.pdf)

[7] Jun Li, Skyler Berg, Mingwei Zhang, Peter Reiher, Tao Wei, DrawBridg e-Software-Defined DDoS-Resistant Traffic Engineering, SIGCOMM'14, pp. 591-592, 2014.

[8] Rishikesh Sahay, Gregory Blanc, Zonghua Zhang, Herve Debar "Towards Autonomic DDoS Mitigation using Software Defined Networking", To be appear, 2015

[9] P. Phaal, S. Panchen, N. McKee, "InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks", IETF, RFC 3176, pp. 1-31, 2001.

[10]Shafqat Ur Rehman, Wang-Cheol Song, Mingoo Kang, "Network-Wide Traffic Visibility in OF@TEIN SDN Testbed using sFlow", Network Operations and Management Symposium (APNOMS), IEEE, pp. 1-6, 2014.

[11] Syed Akbar Mehdi, Junaid Khalid, Syed Ali Khayam, Revisiting traffic anomaly detection using software defined networking, in: RAID'11 Proceedings of the 14th International Conference on Recent Advances in Intrusion Detection, pp. 161-180, 2011

[12] Seyed Mohammad Mousavi, "Early Detection of DDoS Attacks in Software Defined Networks Controller", Thesis submitted to Carleton University, Ottawa, Ontario, 2014

[13] Rodrigo Braga, EdjardMota, Alexandre Passito, Lightweight DDoS flooding attack detection using NOX/OpenFlow, in: LCN '10 Proceedings of the 2010 IEEE 35th Conference on Local, Computer, pp. 408–415, 2010.

[14]S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, "FRESCO: Modular composable security services for software-de fined networks," in Proceedings of Network and Distributed Security Symposium, To be appear, 2013.

[15] Hyojoon Kim, Joshua Reich, Arpit Gupta, Muhammad Shahbaz, Nick Feamster, and Russ Clark. "Kinetic: Verifiable Dynamic Network Control", To appear in USENIX NSDI, 2015.

[16] Erickson, Jon. "Hacking: the art of exploitation" , 2nd ed, No Starch Press, p 256, 2008.

[17] Urs Hoelzle. OpenFlow @ Google. http://opennetsummit.org/archives/apr12/hoelzle-tue-openflow.pdf. Accessed on 02.04.2013.

[18] A.S.Syed Navaz, V.Sangeetha, C.Prabhadevi, Entropy based Anomaly Detection System to Prevent DDoS Attacks in Cloud, International Journal of Computer Applications (0975 – 8887) Volume 62– No.15, January 2013

[19] Marco Canini, Daniele Venzano, Peter Pereˇs´ıni, Dejan Kosti´c, and Jennifer Rexford. A NICE way to test openflow applications. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, NSDI'12, pp. 10–10, Berkeley, CA, USA, 2012.

[20] Amittai Aviram, Sen Hu, Bryan Ford, and Ramakrishna Gummadi. Determinating timing channels in compute clouds. In Proceedings of the 2010 ACM workshop on Cloud computing security workshop, CCSW '10, pp. 103–108, New York, NY, USA, 2010.