SYNTracker: SYN Flooding DoS Attack Defeated by SDN Controller

Ankur Kumar Bajaj[1], Dr. Vineet Richhariya[2]

M.Tech Scholar[1], Professor[2]

Abstract - DoS attacks growing as exclusive chief threats in our whole networking arrangements. SYN requests must be compulsory to establishing the connection but this converts into hazardous when sending in huge amount. This infinite volume of SYN requests for any machine turns it to halting condition which states as Deny-of-Service to all. Since SYN requests are part of traditional environment so it's difficult to change the whole network infrastructure and rules. Therefore, there was need to introduced novel technology in current environment. As a result, SDN and OpenFlow design presented to overcome most of the issues of traditional one. As SDN first decouples the forwarding functionality from control device plane which offered the fast transaction and centralized brain mechanism. To focus on this issue we propose basic procedure named as SYNTracker, which formally targeting number of SYN requests send by particular machine in SDN scenario. Since SDN scenario also offering network infrastructures that resembles to traditional LAN environment with all host connected to one switch. SYNTracker implanted as application on controller which records the SYN requests and when limit crossed then add dropping flow to switch. This dynamic mechanism will supports switch for making decision in attacking position and offers robustness.

Keywords: DoS, SDN, OpenFlow, OpenVSwitch, Mininet, Floodlight Controller, TCP-IP Header.

I. INTRODUCTION

Since independent computer is not adequate to deliver all functionality and information agreeing with our needs. So we interconnected all independent computer to facilitate the services simultaneously that we required to perform specific assignment. But interconnection of computers prerequisite some protocols and infrastructure. Those protocol includes some compulsory procedures to creating connection among the computer devices. These protocol properly needs to put uniqueness among all heterogeneous devices. So TCP/IP stack fulfills the essential demands of networking scenario of collection of connected devices to great scope. Currently the decision take by intermediate device for dispatching packets are done through headers that attached with packets which offered under protocols. But today these traditional protocol becomes potential threat in area of security and performance. Here we discourse the potential vulnerability of TCP protocol which enhance the SYN requests to boundless attack of flooding terms as DoS attack. SDN scenario introduced which resembles to traditional networking environments and supports all classic rules with advance technology of decoupling the forwarding method from control method. The feature of decoupling in SDN scenario facilitate the centralized supervision of network environment. This adds more robustness to our traditional technologies and offers several mechanism to overwhelmed classic protocol weaknesses. The intellect of SDN states as controller has many core features and also offers extra computation through APIs. The API of controller gives developer a realistic power to configure the network through programming. We uses this feature in our study and create the application that configure the switch dynamically after attack accomplished.

This paper propose SYNTracker, a unique SDN application that help to defeat SYN requests flooding Attack in SDN scenario. This also helps the traditional networks in which SDN scenario applied for ease of centralized the local environment. Distinct to allied works, SYNTracker is designed to defeat above issue by introducing dynamic decision for making resolution associated with attack. This dynamic resolution includes flows route related rules, flows timeout, counters resetting dynamically for automation the scenario.

The residual of the paper structured in this fashion. Section II defined the elementary details linked to our propose work. The associated works for mitigating the DoS attacks through various mechanism followed under Section III. Section IV presented the proposed mechanism of SYNTracker in controller and associated setup environment for implanting it through SDN controller module described under Section V. Experimental result and examination are explained through Section VI and finally concluding and indicating future scope with Section VII.

II. ELEMENTARY DETAILS

A) TCP/IP Overview

TCP is ruminate as a trustworthy protocol of TCP/IP suite because it divided data from application level into segments at source and at reception side reassembled them. If data reached at destination not ensure same order as source due to problems in network or packets took different paths for destination then TCP liable for pushing

them in correct sequence. [1] TCP retransmitted a packet when it gets lost while roaming the network, this guaranteed that TCP is trustworthy mechanism for transmission data.

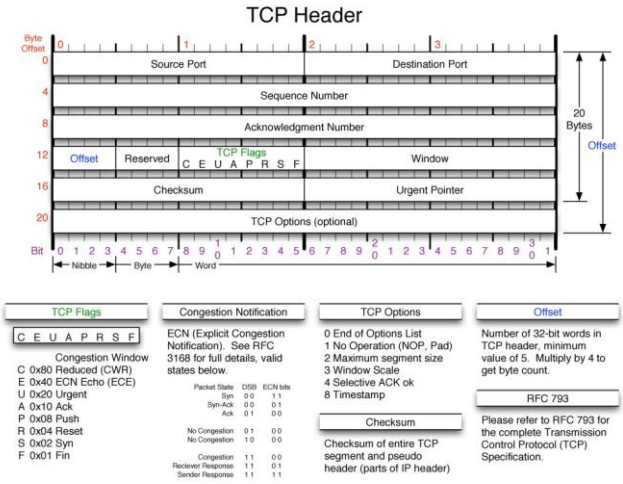

Fig. 1 TCP Header

Application processes resembles to port in terms of networking, sends data through TCP that leads to create a connection among the two devices. Hence ports of the two devices are connected for further consistent transmission of data. This phenomenon states the three-way handshaking concluded for which two device are orchestrated the communication channel for exchange data. To start this mechanism first sender sends the TCP packet with sets SYN flag to 1 in TCP header of first segment. Then receiver send TCP packet with sets SYN flag to 1 and ACK flag to 1 in second segment in accordance to accept the first segment. Finally sender sends TCP packet with sets ACK flag to 1 in TCP header of final segment for accomplishing synchronization. The TCP packets used flags with sets to 1 them as prerequisite for definite purpose which resolves to hexa-decimal values for presenting the header clarified in Fig 1.

B) SDN Overview

Software-Defined Networking is an evolving design which facilitate dynamic, controllable, cost-friendly, and compliant functionality to recent applications. This only attainable through physical bifurcation of the control tier from the forwarding tier for centralized supervision [2]. The centralized architecture provides the configuration of switches and controlling flows through programming by means of OpenFlow protocol. SDN controller work as a centralized device that configures, maintain and optimize the network resources.

Fig 2 illustrated the classic architecture of SDN as distributed into three fragments: Application plane, Controller plane and Infrastructure plane. Applications are the programs that provides services to network resources for making decision on specific logic through a northbound API. Northbound states the interface among Application plane and Controller plane that explicitly conveys the services. Controller plane works as the mediator and centralized service provider among upper tier and lower tier. SDN Controller provides the forwarding decision to infrastructure plane devices from translating upper level application program through southbound API. Similarly it provides the précised vision of network of infrastructure plane to top layer programs. Infrastructure plane resides of numerous varieties of switches either physical or based on hypervisor which connects the all devices for communication. These switches typically controlled by controller through OpenFlow interface for making decision on forwarding flows.

[2] OpenFlow denotes as the typical interface among controller and infrastructure plane's switches of SDN scenario. OpenFlow is chief protocol that facilitate the straight access to network devices of infrastructure plane as it recognize by network devices and also SDN controller. OpenFlow works on perception of flow to recognize network load on basis of predefined match rules which may be statically or dynamically configured by SDN controller. Enterprises and ISP ease to introduce OpenFlow involved SDN technologies as network devices friendly support OpenFlow and traditional forwarding concurrently.

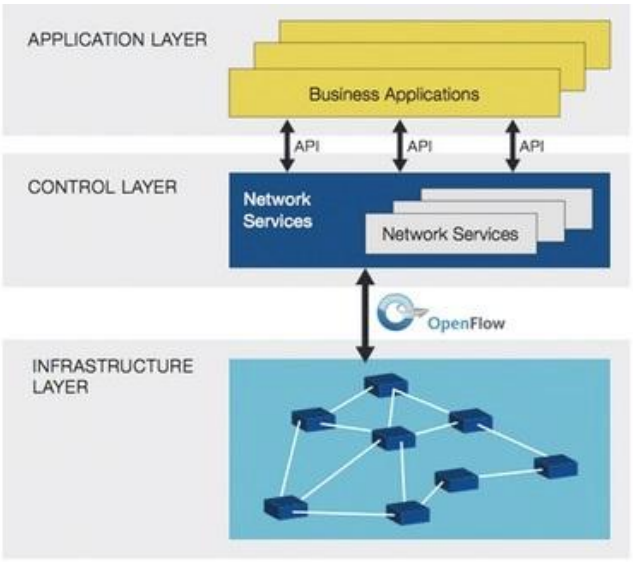

Fig. 2 SDN Architecture

C) Mininet

It is a tool whose functioning resemble to network emulator that builds a network of virtual hosts, switches, controllers, and links. Virtual hosts that created under emulation runs classic Linux network software. Its

switches supports OpenFlow technology to provide tradition forwarding and also new SDN environment. [3] Mininet can supports research investigation such as building, testing, and more tasks in networking on single laptop or PC. It typically creates a complete network which running the hosting system kernel to comply virtualization mechanism in lightweight routine. Virtualization concept facilitate the virtual host behaves identical to real system and run arbitrary command of Linux system installed on hosting machine.

Mininet become favorable in research purpose as it provides many features for network system. The features which made it popular in researchers and also in developers are:

- It is simple and fast to initializing a network in few seconds that resolves a loop of run-debug-edit in quick way.
- It facilitate to create topologies in custom approach which resembles to a datacenter, a backbone, huge LAN etc.
- It provides ability to run linux based command and programs on its virtual hosts, from web-browsers to monitoring tool like Wireshark.
- In mininet, packet forwarding also be customized using programming through OpenFlow protocol in switches.
- Mininet experiments can easily build and execute using writing simple Python scripts.
- Sharing and reproduce results of scripts on every computer as it packaged once.

Using little command can form up a classic network of switches connected to several hosts and a single Controller, as presented in Fig. 3.

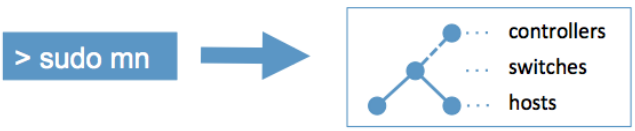

Fig. 3 Mininet Example

D) Floodlight Controller

[4] Floodlight project is open-source built on Java platform and licensed under Apache.

It is belongs to Open SDN controller family founded by Big Switch. It works on OpenFlow protocol mechanism in a SDN environment to regulate forwarding flows of switch. It facilitate to work with physical switches like Cisco and hypervisor switches like OpenVSwitch as configured to OpenFlow protocol. It has influence designed to delivered functionality to vast collection of physical switches and switches based on hypervisor. Demonstrated the scenario using Fig. 4 that forwarding plane is controlled by the chief control tier with supporting provided by application tier modules. Applications designated under upper tier delivers new services to core controller for enhancing the functionality. It is grounded on modular platform includes module loading system through which easy to extend with adding new modules to it. Modular platform also provided secure implementation of core modules so they couldn't exposed to upper tier applications.

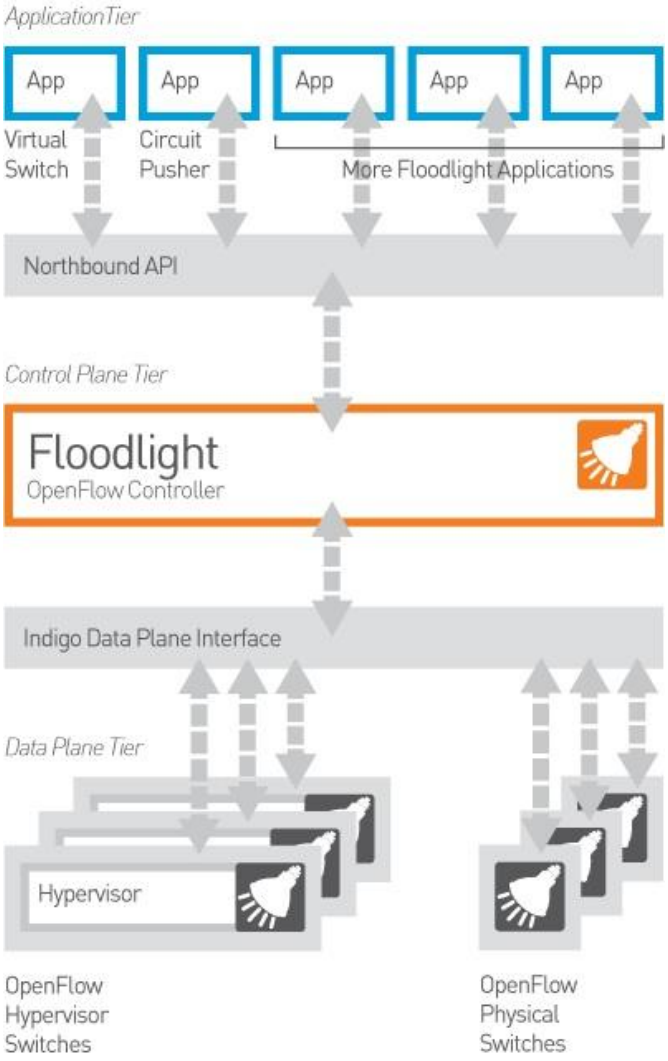

Fig. 4 Floodlight Controller

III. RELATED WORK

To defend the network services from various attacks, several techniques have been suggested by many researchers. As our data and network resources are become more valuable entity for users and also to their providers. But these things has various weakness as they are created by humans therefore vulnerabilities utilized by other humans say them attacker or disruptors.

DoS attacks are one of vital issue that arises from these weakness of human created infrastructures and mechanisms. DoS attacks has several kinds but here we are focused on flooding grounded strategy attack. SYN

flooding is one of DoS attacks which utilized the vulnerabilities associated to TCP/IP suite defined for synchronizing mechanism among two machines. Most of the researchers recommended fusion of traditional networking with SDN scenario to defeat these attacks to great level.

Firewall stands in better way for resolution in defeating several attacks mostly DoS attacks. As firewall also implemented in SDN atmosphere as an application works on top level. [5] Recommended to utilize firewall as application in SDN scenario for better resolution in term of security and also utilize load balancing for performance issues. But suggested firewall application has some issues as it doesn't support dynamic decision. It supports only proactive rules which states configure the firewall guidelines in advance.

Kuerban et al. [6] suggested new strategy stated as FlowSec, for mitigation of DoS attack targeted the controller implanted under SDN scenario. Formally, FlowSec defined mechanism to control frequency of packets sends to controller as specified under OpenFlow scenario. It mainly focused on volume of packets message transaction among controller and switch for different purposes. So it used the meter function to control bandwidth of control layer of SDN scenario. Similarly, Wei et al. [7] suggested FlowRanger algorithm to defend the controller from unnecessary floods and serves only legitimate traffic to controller. For performance improving, they used queueing functionality with priority in their module which implemented at controller side. So the trusted packet based on source arranged in that queue with trust value as priority and then processed by controller through traditional scheduling. But these strategies also doesn't defined the dynamic functionality to regulate flooding of packets.

As far from above work, Konidis et al. [8] proposed different functionality of redirection of TCP packets using SDN scenario. Some slightly changes done in core functionality of SDN to accomplish their task of redirection. As they changed the traditional format of request sent from client to server for distinguished the new format. They claimed their strategy as fast in tendency of performance of controller to redirect the requests arrived from user to backup server.

Dao et al. [9] presented a resolution to guard SDN scenario from Distributed DoS attacks through IP tracking mechanism. The resolution stated the analysis of services utilized by user and their behavior to calculate timeout of flow associated to them. But this strategy creates problem, sometimes drop all malicious traffic, which might be false positive flows.

Dridi and Zhani [10] described novel methodology to defeat DoS attack in SDN scenario through decision making strategy. As they configured three module: Flow management, rule aggregation and monitoring of flows. These modules works in independently and creates a decision making table for distinguishing among legitimate and malicious packet.

None of the above methodologies are capable to take decision dynamically to regulate the flooding of packets regarding SYN requests to selected machine like server. So leading to this way, we propose a solution of tracking a SYN requests to any machine from others. After tracking, decision will take to regulate specified machine from utilizing services that grounded on TCP.

IV. PROPOSED METHODOLOGY

This section defined the proposed design of SYNTracker mechanism to diminish SYN flood packets originated by attacker for target server. For attaining our aim, there are some prerequisite configuration on traditional SDN environment.

This includes basic component of OpenFlow protocol such as Flow table, Flow entry, Flow match field, Action. The switch, supports OpenFlow, contains either one or many flow tables which subjected to version of OpenFlow protocol used, to perform packets matching and associate forwarding. The switch managed by controller through OpenFlow protocol that empowered to add, alter and delete of entries from switch flow tables. This customization done in two ways, first by reactively states in response of packet and second by proactively states manual way.

Formerly a packet received at switch for forwarding, first matching functionality started in flow tables for comparing it to entries. If matching entry found in entries of flow tables then accomplish action associated for that entry. If no matching found for receive packet then usually switch send it to controller for appropriate action. This forwarding of packet originated by switch done through OpenFlow channel to controller separated from local network. This scenario typically resides in types of asynchronous messages of OpenFlow messages sustained under protocol. This asynchronous messaging are originated from switch to controller for updating network changes to controller. This asynchronous messages has four types Packet-In, Packet-Out, Flow Removed and Port status.

Functionality of Packets-In messages utilized by switch while no match entry found under entries of flow tables for received packet. Then switch redirects the received packet to controller by wrapping up in Packet-In message.

After reception of Packet-In message by controller, an analysis begins then appropriate configuration sent to switch regarding to same.

In our methodology, illustrated by fig 5 using flow chart, we utilize functionality of Packet-In message accomplished among the switch and controller. We first add a flow rule in flow table of switch regarding all TCP packets are first sent to controller after arriving at switch. Now our SYNTracker module implemented over controller checks all Packet-In message. If TCP packets has flag set to SYN then obtained MAC of source from packet header. And we organized a map of MAC to SYN counts stored in our module to record the track of SYN counts associated from that specific MAC. So SYNTracker increase the count by 1 if flag set to SYN of received Packet-In message for associated MAC in map. Here checking of map of MAC to SYN counts done when incrementing the count that if it crossed the limit which we assigned under SYNTracker.

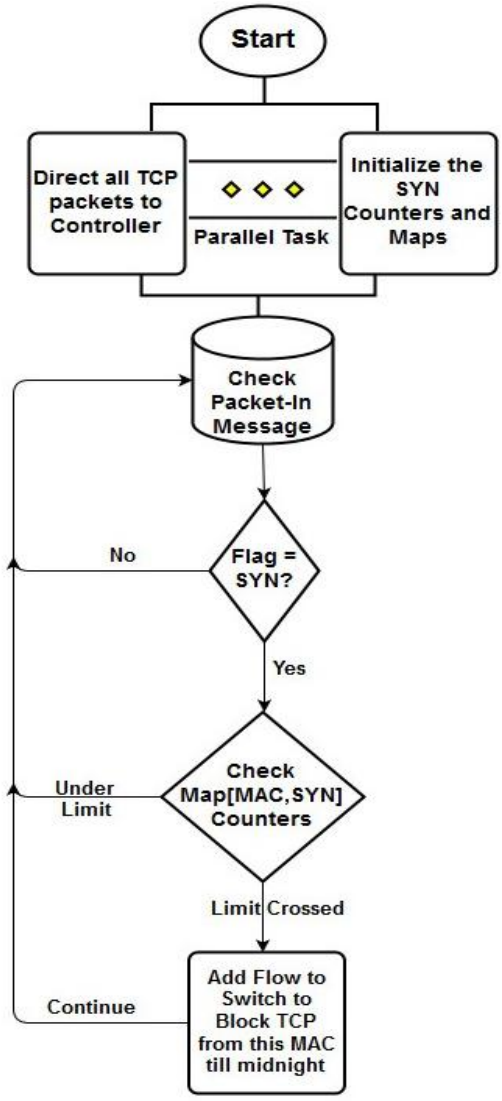

Fig. 5 Flow Chart of Proposed Solution

If limit crossed by some MAC then SYNTracker utilized the flow manipulation strategy offered by OpenFlow protocol through controller and add the flow rule to switch. The flow rule contains action to drop regarding same MAC with high priority than existing flow entries. Now if TCP packet reached that has flag set with SYN from same MAC then switch matched with drop rule and drop it. But also includes one feature supported by flow rule of time-out functionality. When controller added a flow of drop, time is calculated in seconds till midnight for time-out field of flow rule. Flow rule automatically deleted after time-out reached and switch will informed controller about it.

V. EXPERIMENTAL SETUP

This segment defined design of experimental arrangement used to emulate an SDN scenario and tool to generate SYN flooding DoS attack traffic.

The experiment conducted through machine which configured as processor of Intel(R) Core(TM) i3-5005 CPU of frequency 2.00GHz with 2 cores resembles to 4 cores with RAM of 8GB. Ubuntu 14.04 used to run java based floodlight project for light weight processing of operating system. Mininet emulator used for creation of network topology that generates virtual hosts on the hosting platform. Mininet also creates OpenVSwitch based switches and connections among the virtual hosts to switch. It also facilitate choosing of controller through configuration of controller's address in switch.

Here we used OpenFlow controller for controlling the switches of SDN that is Floodlight master based on java platform. OpenFlow protocol ver 1.3 used as interface medium among the switches and floodlight to synchronize the further communication.

Topology Scenario: The experimental emulation design resided 4 hosts and switch which revealed through fig 6.

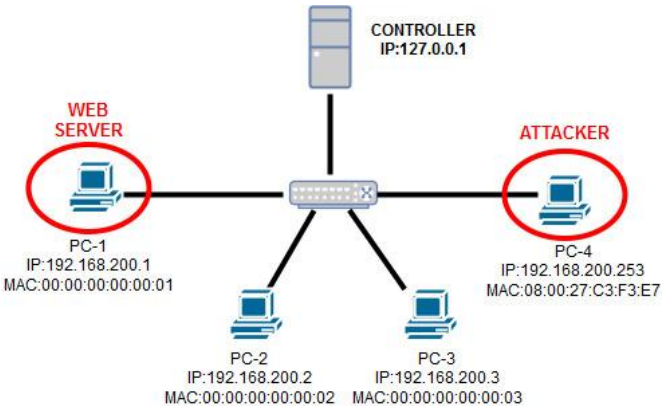

Fig. 6 Experiment Setup

Here the switch connected to controller which not part of local network through secured physical port.

Fig 7 presented the mininet tool command to emulate the above scenario of experiment.

- The remote option used to connect the explicit controller instead of built-in.
- The OpenFlow13 declared the protocol version of OpenFlow
- The ipbase used to modified the range of IP in local network for experiment.
- HTTP server configured on PC 1 using simple command facilitate by mininet.

Fig. 7 Mininet Command

SYN Flood scenario: To generation SYN flood we used the hping3 through PC4 in experiment. Here the PC1 acts as simple web server and PC4 generates SYN flood to server. [11] Scenario of SYN Flood generated by hping3 with help of Kali Linux to PC1 illustrated by fig 8. Here hping3 used port #80 for http from different source using random option in flooding approach. Flood method used to require no replies from target server. Every single packet resides data of 128 bytes which turns target in busy situation.

Fig. 8 Hping3 Command

The flooded packets forwarded by switch which redirect the TCP load to controller. For our proposed solution prerequisite one assumption states redirect all TCP packets to controller. Hping3 generated the TCP packets sets SYN flag to 1 with flood option to target server.

VI. EVALUATIONS AND RESULT

Our evaluations strategy starts from the TCP packets header values to obtained preview of attack regarding SYN flooding. First we use the great tool of monitoring the traffic, Wireshark [12], for obtaining screenshot the header values of any TCP packet. Fig 9 illustrated the screenshot of TCP flags of particular packet which heading towards the flags values showing in hexadecimal format and briefly describe that SYN is set.

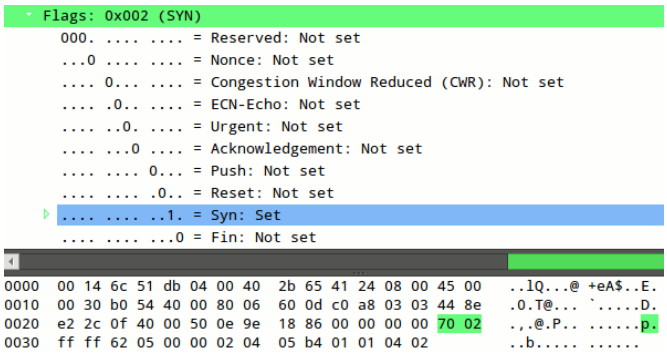

Fig. 9 Wireshark Screenshot

According to above scenario of analysis the header values of TCP through packet investigation, our mechanism focused on flags values. Therefore SYNTracker module first applied a rule in switch table to redirection of packets regarding to TCP to controller. Fig 10 described the flow entries screenshot obtained by show command of OpenVSwitch that TCP are first send to controller.

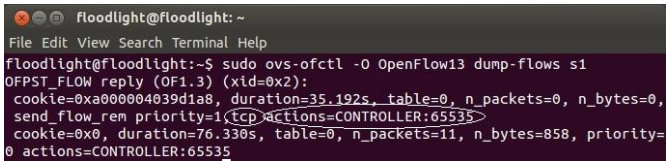

Fig. 10 Switch Table Screenshot

After configuring this rule, every TCP arrived at switch first sends to controller in form of Packet-In then some appropriated action performed. Now SYNTracker module utilized the core features of floodlight to obtaining the payload of arrived Packet-In. Then under SYNTracker module, a map of MAC to SYN counts stored using MAP feature of java. This map basically used for collection of values and checking of SYN counts of particular MAC. After crossing the limit that we set in beginning of SYNTracker, the new rule will add to switch. Fig 11 presented the screenshot when new rule of dropping packets of any MAC applied to switch table.

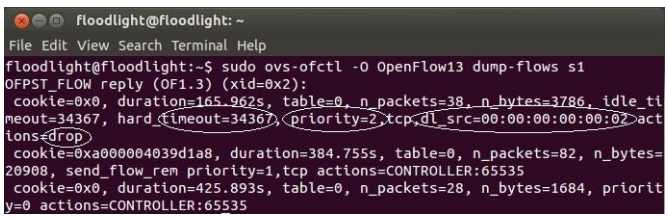

Fig. 11 Drop Rule Screenshot

The rule includes the time-out of rule, priority and source MAC regarding to particular system.

According to our investigation, we used kali system to create scenario of attack on web server that we configured on virtual host using mininet. As kali system (stated as PC4) used the hping3 tool to create fictitious packets that

includes SYN request to our webserver (stated as PC1). PC1 organized with IP [192.168.200.1], MAC [00:00:00:00:00:01] and PC4 with IP [192.168.200.253], MAC [08:00:27:C3:F3:E7]. When configured limit of SYN counts crossed which is tracked by SYNTracker module then new dropping rule against PC4 applied to switch table, as presented by fig 12. Figure stated that drop the packet that has MAC [08:00:27:C3:F3:E7] in source field of header.

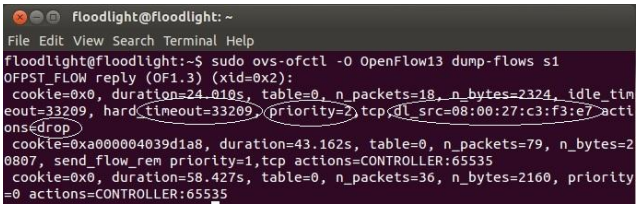

Fig. 12 Drop Rule Against PC4

After applied this rule of dropping, only TCP packets are dropped of particular MAC. As our rule applied on TCP, every packets is allowed to forward through switch from PC4 except TCP. As presented in fig 13, that PC4 allowed to direct ICMP to our web server but couldn't connect to services that utilized TCP.

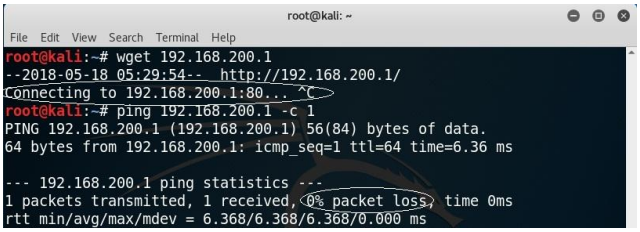

Fig. 13 Allow Other But No TCP

To study the behavior of our web server (stated as PC1), we examined the traffic arrived on it through Wireshark tool then plotted the graph. Using the feature of IO Graph under Wireshark, we first analyzed the behavior of web server before our SYNTracker module implanted on controller presented by fig 14.

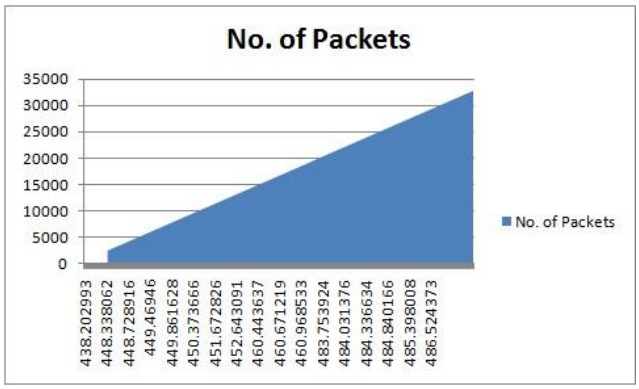

Fig. 14 Load Before SYNTracker

After SYNTracker implemented on controller, the behavior suddenly changed as all TCP dropped on switch level and never arrived presented by fig 15.

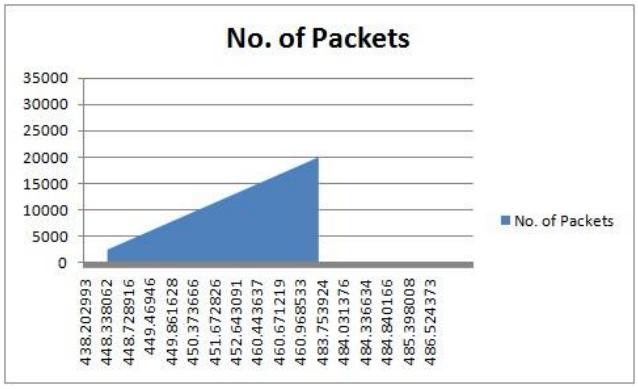

Fig. 15 Load After SYNTracker

As examination on two graph indicated that the TCP packets includes with SYN request are resolved at switch level due to our SYNTracker module implemented on controller. This enhances the advantage of dropping TCP packets that includes with SYN request after some limit that we investigated using human behavior. So every machine allowed to utilize the services based on TCP and ICMP protocol but under some boundary.

VII. CONCLUSION AND FUTURE SCOPES

Our paper proposed a solution named SYNTracker to defeat the SYN flooding grounded DoS attack originated on any server. Regarding to SYNTracker, first tracking the TCP packets with SYN request and count them to check the limit we allowed. So when any machine crossed the limit, dropping rule will applied to switch which doing forwarding functionality of packets for that machine.

In future SYNTracker module will implemented in infrastructures of cloud scenario as controller connected to switches grounded on hypervisors and create SDN environment under Cloud. Then virtual machine created under cloud connected through switch, will guarded from SYN based attack.

As referred to other schemes in this domain advantage of our SYNTracker module is to allowing every form of packets under limit. If limit crossed by any machine then blocked it for specific time using time-out feature of rule. After elapsing of specific time, allow that machine to utilize the services in network. But due to fully dependency on controller, as every TCP packets travelled to controller, burden of traffic slow down its mechanism. This will overwhelm when only those TCP packets are directed to controller which has SYN request rule configured on switch. This facility only offered by OpenFlow ver 1.5 as they provided the rule which allow the matching of flags values also. When floodlight

controller will supports higher versions like OpenFlow ver 1.5 and above then this problem of maximum load will reduced to great scope.

REFERENCES

[1] "Characteristics of TCP." http://www.omnisecu.com/tcpip. [Online; Accessed: 11-May-2018].
[2] C. Fernandez and J. L. Muñoz, "Software Defined Networking (SDN) with OpenFlow 1.3, Open vSwitch and Ryu," thesis.
[3] "Mininet." http://mininet.org/. [Online; Accessed: 11-May-2018].
[4] "Project Floodlight" http://www.projectfloodlight.org/. [Online; Accessed: 11-May-2018].
[5] N. Zope, S. Pawar, and Z. Saquib, "Firewall and load balancing as an application of SDN," CASP, 2016.
[6] M. Kuerban, Y. Tian, Q. Yang, Y. Jia, B. Huebert, and D. Poss, "FlowSec: DOS Attack Mitigation Strategy on SDN Controller," IEEE International Conference on Networking, Architecture and Storage (NAS), 2016.
[7] L. Wei and C. Fung, "FlowRanger: A request prioritizing algorithm for controller DoS attacks in software defined networks," IEEE International Conference on Communications (ICC), 2015.
[8] E. Konidis, P. Kokkinos, and E. Varvarigos, "Evaluating Traffic Redirection Mechanisms for High Availability Servers," IEEE Globecom Workshops (GC Wkshps), 2016.
[9] N.-N. Dao, J. Park, M. Park, and S. Cho, "A feasible method to combat against DDoS attack in SDN network," International Conference on Information Networking (ICOIN), 2015.
[10] L. Dridi and M. F. Zhani, "SDN-Guard: DoS Attacks Mitigation in SDN Networks," 5th IEEE International Conference on Cloud Networking (Cloudnet), 2016.
[11] "Hping3: Penetration Testing Tools," https://tools.kali.org/. [Online; Accessed: 14-May-2018].
[12] "Wireshark." https://www.wireshark.org/. [Online; Accessed: 16-May-2018].