Journal of Advances in Science and Technology
Vol. 12, Issue No. 25, (Special Issue) December-2016, ISSN 2230-9659

Prevention of DDoS Attacks in SDN by Using Virtual IP Addresses

Sanjeetha R.[1]*, Monisha B.[2], Anita Kanavalli[3]

Department of Computer Science & Engineering, M.S. Ramaiah Institute of Technology

Abstract – DDoS is one of the most common attack that is prevalent in traditional networks, it also has its implications on Software Defined Networks (SDN). SDN is a new network architecture which separates the data plane from the control plane. In traditional networks an attack is performed by first identifying the IP addresses of the victim machine and then sending huge amounts of unnecessary data to it. A similar attack can also be performed on servers that are present in SDN. In our paper we propose a method wherein the real ipaddress of important servers can be hidden thereby preventing the DDoS attack. A DDoS Detection and Prevention modules are deployed on the SDN controller. The DDoS detection module identifies that there is a DDoS attack and differentiates legitimate clients from botnets. The DDoS prevention module generates virtual ipaddresses for every real ipaddress dynamically that changes regularly after some interval. The SDN controller makes use of the results of these two modules and installs rules into flow table such that only the legitimate clients will be provided with the real ipaddress whereas the botnets are blocked by dropping their requests.

Keywords— SDN, DDoS, Real Ipaddress, Virtual Ipaddress, SDN Controller, DNS

----------♦----------

1. INTRODUCTION

SDN is a new network architecture that separates the data plane from the control plane. The SDN controller implements the control plane and is responsible for making forwarding decisions and installing rules in the flow table. The network devices like hubs, switches and routers implements the data plane. They simply forward the packets based on the rules installed in their flow table. SDN can be implemented in many ways viz. , SDN via APIs, SDN via hypervisor based overlay networks and Open SDN. The vendors of the existing traditional switches provides APIs to their devices through which SDN can be implemented, however complete control is not given to the users, and they still need to use specific APIs provided by those vendors this is called as SDN via APIs. The underlying physical network can be kept intact and an overlay network can be created using tunnels that implements SDN, but this method tends to have more overhead on the network, this is called SDN via hypervisor based overlay network. Open SDN actually implements the concept of SDN by providing full control of network to the users. It is implemented using open Flow switches and controllers like NOX, POX, flood light etc. In our paper we implement the proposed solution using Open SDN. [1]

Similar to the traditional network, SDN also uses DNS for hostname to ipaddress translation. The client sends a query to the DNS to resolve the server hostname and the DNS replies it with the corresponding ipaddress. The client later uses it as the destination ipaddress, generates the packet and sends it to the forwarding switch to which it is connected The DNS in SDN can be implemented on the controller or as a separate server..

The working of switches in an Open SDN is as follows

1. When a switch receives a packet through its input port, it checks its flow table for any entries that matches the fields in the incoming packet like ipaddress, mac address, port number etc., if a match is found it simply performs the actions specified in the matching rule which includes forwarding the packet on to the specified output port or dropping the packet. If no match is found, the packet is immediately forwarded to the controller.

2. When the controller receives the unmatched packet from the switch, it decides as to how the packet should be handled using different algorithms, applications or protocols. The controller then creates a new flow rule for this packet and installs this rule on the requesting switch and all other switches under its

control. The unmatched packet is sent then back to the switch.

3. The switch now refers to the new rule and performs the specified action on that packet. [1]

IMPLEMENTATION

Modules Description

SDN Controller Module – This module works as the control plane of the SDN. It is responsible for installing rules into the flow table in the forwarding switches.

Forwarding switch Module – This module receives the packets on its input port, checks the flow table for a matching entry and takes the actions specified. If no matching is found it forwards the packet to the controller.

Host machine modules – These modules implement the sender which sends requests to the server. All the packets from hosts are sent through forwarding switch. If the hosts send packets within specified threshold they are considered to be legitimate clients, if the sent packets exceed the threshold they are called botnets.

DNS Module – This module is implemented on SDN controller. It translates hostname of the server to its corresponding ipaddress and sends the result to the DDoS prevention Module.

DDoS Identification Module – This module is implemented on the SDN controller. It observes the amount of traffic generated by each host and differentiates the legitimate clients from botnets. If the traffic generated by the host is within the set threshold, it is identified as an illegitimate client otherwise it is considered to be a botnet. The list of legitimate client and botnet is sent to the SDN controller module for further action. [2]

DDoS Prevention Module – This module is implemented on the SDN controller. It receives real ipaddress from DNS. It randomly and dynamically generates a virtual ipaddress for each real ipaddress and maintains this information in a rIP-to-vIP mapping table. Further it sends the virtual ipaddress and the real ipaddress to the SDN controller. [3][4][5]

2. WORKING PROCEDURE

When a host machine i.e client/botnet requests for hostname translation of the server, the request is sent to DNS through the forwarding switch. The DNS performs the mapping and sends the results (the real ipaddress) to DDoS prevention module. The DDoS prevention module randomly generates a virtual ipaddress for the real ipaddress and maintains this information in a rIP-to-vIP mapping table. The virtual ipaddress and its corresponding real ipaddress is then sent to the SDN controller. The controller then sends virtual ipaddress to all the requesting hosts through the forwarding switches.

Meanwhile the DDoS detection module runs and observes the traffic generated by all the hosts and based on a predefined threshold value classifies the host machine into legitimate clients or botnets. The list of clients and botnets are provided to the SDN controller and the DNS.

The SDN controller then forwards the real ipaddress with the corresponding virtual ipaddress to the forwarding switch and adds flow rules such that the legitimate clients are provided with the real ipaddress and the botnets continue with the virtual ipaddress. By doing so the legitimate clients will be able to reach the server, while the packets from botnets are dropped as there is no server with the specified virtual ipaddress.

Further the botnet may retry requesting DNS for the translation again, but the DNS will block such requests by dropping the packets generated from them.

In order to make sure that the botnet does not get hold of the virtual ipaddress to real ipaddress mapping, the virtual ipaddresses are dynamically changed after some specified time interval.

By implementing this method the botnet is not able to find the real ipaddress of the targeted server and is thus not able to perform an attack.

3. EXPERIMENTAL RESULTS

The proposed solution is implemented using MATLAB simulator and the results observed are as shown below. We have considered Four servers as the target or victim servers on which a DDoS attack is planned. The hosts colored blue are legitimate clients which generate traffic within specified threshold, the hosts colored red are botnets which generate unnecessary traffic exceeding the threshold. For experiment purpose we have generated traffic using ping requests from all the hosts, and the servers send ping replies only to the legitimate clients.

Figure 1 shows the SDN setup with virtual ipaddress given to all four servers for the first time. The blue colored line in the graph show the packets that are sent by legitimate client which slowly increase over a period of time. Initially the packets are dropped because they are not yet provided with the real ipaddress. Once the hosts are identified as legitimate, they are given real ipaddress and continue sending packets without any disruption. The red colored line shows the number of packets sent by the botnets which increases very quickly as they are sending more packets to perform attack. But at one point they are stopped as their packets are dropped because of non availability of real ipaddress. They further continue trying again which can be observed with the

increase in packets sent by them. Figure 3 shows the SDN scenario after the botnets request for translation the second time. Observe that this time the number of botnets sending the packets are reduced as many of them are already identified as botnets and their requests are dropped. This process continues and after few iterations the requests from botnets are completely blocked and the packets sent by them drops drastically. The green colored line in the graph shows the total traffic in SDN.

Figure 2 and 4 shows the flow rules installed in the flow table. It shows the mapping of virtual ip with real ip and gives the details (i.e ipaddress) of hosts that received the real ipaddress.

Throughout this experiment the botnet is assumed to be static i.e the same set of host machines are considered to be the part of botnet.

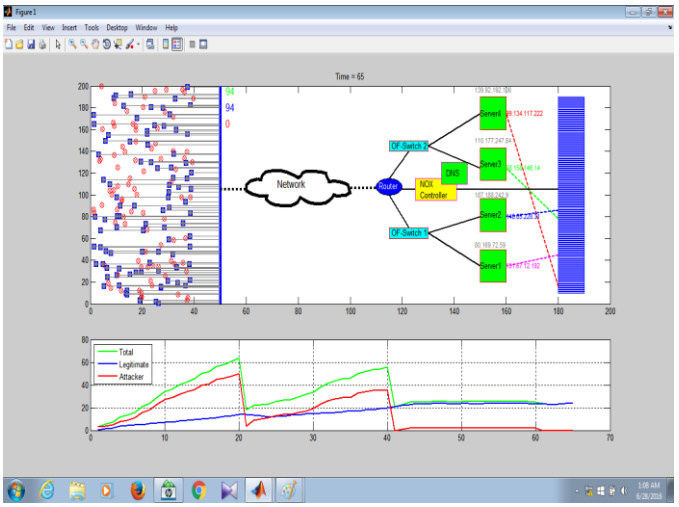

Figure 3: SDN after providing virtual IP address for the second time

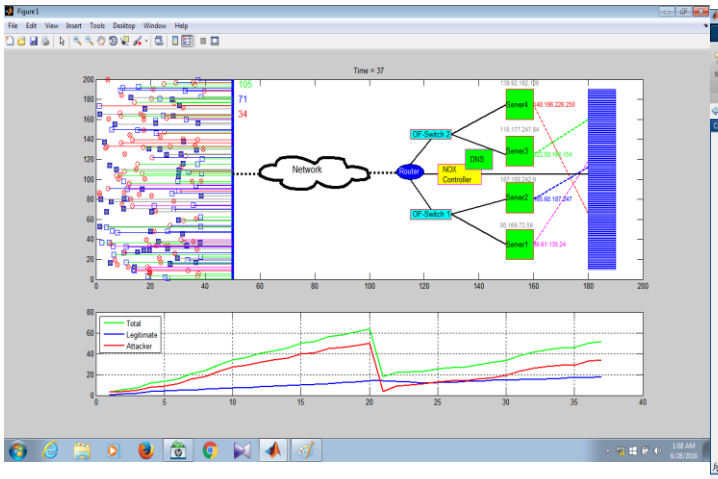

Figure 1: SDN after providing virtual IP address for the first time

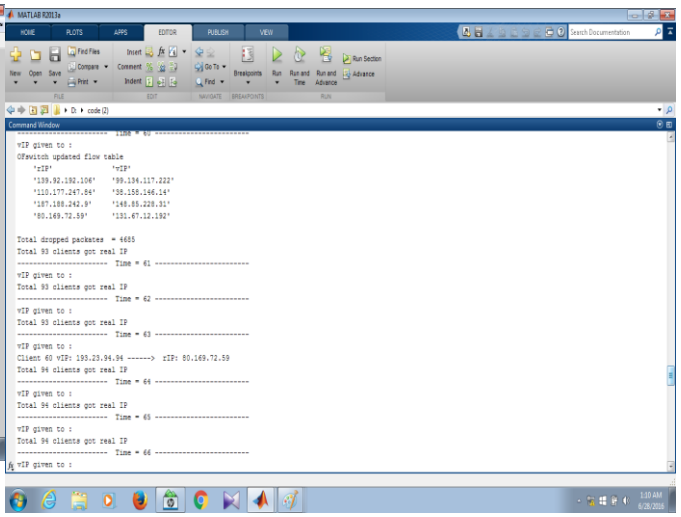

Figure 4: Snapshot of Open Flow Switch table for second set of virtual ipaddress

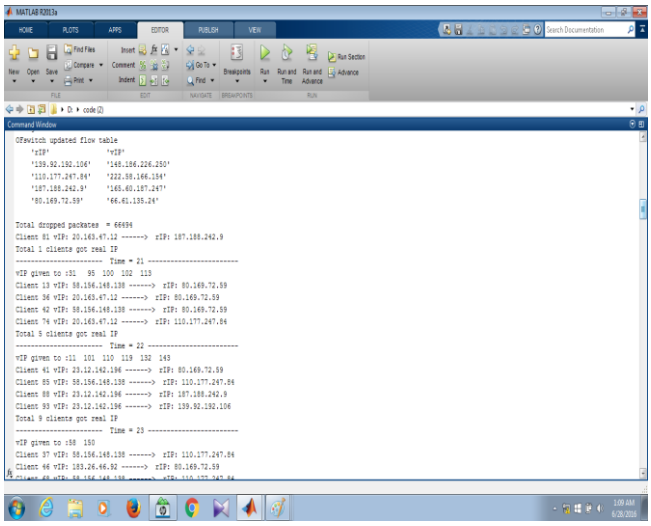

Figure 2: Snapshot of Open Flow Switch table for first set of virtual ipaddress

4. CONCLUSION AND FUTURE SCOPE

In this paper we are able to prevent attack on important applications/servers in a SDN environment by providing virtual IP address initially through SDN controllers to all the requesting clients. Later the legitimate clients are identified and provided with real IP address and botnets are blocked by dropping their packets.

The solution can be further extended to prevent DDoS when the set of host machines that form a part of botnet become dynamic i.e new host machines become a part of botnet and existing ones leave.

5. ACKNOWLEDGMENT

We acknowledge the support and help provided by HOD, Department of Computer Science &

Engineering, Principal and Management of M.S. Ramaiah Institute of Technology, Bengaluru (an autonomous college affiliated to VTU).

6. REFERENCES

Paul Göransson Chuck Black, "Software Defined Networks A Comprehensive Approach", Morgan Kaufmann is an imprint of Elsevier 225 Wyman Street, Waltham, MA 02451, USA Copyright © 2014 Elsevier Inc.

Yu Chen, Kai Hwang, Fe and Wei-Shinn Ku, "Collaborative Detection of DDoS Attacks over Multiple Network Domains", Manuscript received August 14, 2006; revised Dec. 23, 2006; accepted April 10, 2007; published online June 2007. Recommended for acceptance by S. Olariu.

Jafar Haadi Jafarian, Ehab Al-Shaer, Qi Duan, Kijoon Chae, "OpenFlow Random Host Mutation: Transparent Moving Target Defense using Software Defined Networking", HotSDN'12, August 13, 2012, Helsinki, Finland. Copyright 2012 ACM 978-1-4503-1477-0/12/08 ...$15.00.

Sandeep Singh, R.A. Khan and Alka Agrawal, "Prevention Mechanism for Infrastructure based Denial-of-Service attack over Software Defined Network", ISBN:978-1-4799-8890-7/15/$31.00 ©2015 IEEE

Mark Shtern, Roni Sandel, Marin Litoiu, Chris Bachalo, Vasileios Theodorou, "Towards Mitigation of Low and Slow Application DDoS Attacks". DOI 10.1109/IC2E.2014.38

Corresponding Author

Sanjeetha R.*

Department of Computer Science & Engineering, M.S. Ramaiah Institute of Technology

E-Mail – sanjeetha.r@msrit.edu