Leveraging SDN for Collaborative DDoS Mitigation

Sufian Hameed, Hassan Ahmed Khan
IT Security Labs, National University of Computer and Emerging Sciences (FAST-NUCES), Pakistan
sufian.hameed@nu.edu.pk

***Abstract*—In this paper we propose a collaborative distributed denial of service (DDoS) attack mitigation scheme using SDN. We design a secure controller-to-controller (C-to-C) protocol that allows SDN-controllers lying in different autonomous systems (AS) to securely communicate and transfer attack information with each other. This enables efficient notification along the path of an ongoing attack and effective filtering of traffic near the source of attack, thus saving valuable time and network resources. We developed and deployed a prototype of the proposed scheme in our lab to evaluate the performance and efficiency. Based on the experimental results we showed that our SDN based collaborative scheme is capable of efficiently mitigating DDoS attacks in real time with very small computational footprints.**

***Index Terms*—DDoS, SDN, Software Defined Networking, Software Defined Security**

I. Introduction

DDoS attacks are here since the very advent of computer networks and they are not going anywhere anytime soon. Recently, IoT devices (such as printers, cameras, home routers and baby monitors) were used to generate DDoS attack involving malicious DNS lookup requests from tens of millions of IP addresses [1]. This attack is considered largest of its kind in the history with an unprecedented rate of 1.2 Tbps. The main target of the attack were the servers of Dyn, a company that controls much of the Internets domain name system (DNS) infrastructure.

Popular defense practice against DDoS is to deploy detection and response mechanisms at the destination hosts due to higher accuracy and cheaper cost. On the downside destination based mechanisms alone cannot mitigate attack on the paths to the victim and waste resources. This calls for an efficient mitigation strategy to ease out network resources along the transit path of an attack from source to victim.

SDN bring us new approaches to deal with DDoS attacks [7]–[10], [13]–[15]. The separation of control and data plane in SDN allows us to write the control logic and instruct the forwarding plane to behave accordingly. This programmability gives us more control of the network traffic which was not possible before the advent of SDN. In [11], Giotis et al. proposed a DDoS mitigation scheme across multiple SDN domains or networks[1]. The mitigation process starts from the victim network and propagates along the way towards the source. They extended the BGP protocol to embed the incident report as URIs within BGP signals. This reliance on BGP have some ramifications. First of all BGP is very complex and hard to master, and any modifications to existing protocol will challenge the deployment. Secondly, the exchange of incident report between adjacent domains is not instantaneous and will only take place after every BGP update interval. Therefore, the report latency will increase with the number of hops between the source and victim of attacks. Further, they do not validate the authenticity of incident reports exchanged among the adjacent SDN domains. This could make the whole infrastructure vulnerable to fake incident reports from malicious domains.

In this paper, we propose a lightweight, efficient and easy to deploy collaborative DDoS mitigation scheme leveraging SDN. We have design a secure C-to-C communication protocol for SDN-controllers lying in different autonomous systems (AS). This allows SDN controllers to effectively communicate with other controllers in the neighbouring domains and inform them about an ongoing attack. Through this approach the SDN controllers are able to simultaneously block the malicious flows within the network and inform the neighboring domains/networks about an ongoing attack.

This way we are not only able to successfully mitigate the DDoS attack within the victims's network but the transmission of attack information along the path of an attack (transit networks) enable us to filter the DDoS attack close to the attack sources. This results in preserving of valuable network resources along the attack transit path.

Although push-back schemes to mitigate DDoS attack along the attack path has been discussed in the research community [16], [17], but they require more resources at various levels and the push-back mechanism must be deployed in all the participating network components (routers and switches). The complexity and overhead because of the coordination and communication among distributed components adds serious management challenges. SDN based deployments on the other hand ease the management challenges, where a single controller can manage the coordination among all the network components at the AS level. The proposed C-to-C communication protocol is flexible and it can be easily appended with best know DDoS detection engines. Further, the protocol itself can use different approaches for deployment. It can be deployed in linear order, peer-to-peer or via centralized scheme to collaboratively disseminate DDoS filtering information.

In order to assess our proposed collaborative DDoS mitigation scheme we deployed prototype testbeds in our laboratory. The evaluation results are quite promising and demonstrate the effectiveness, flexibility and scalability of the proposed approach.

[1]Domain(s) and Network(s) are used interchangeably throughout this paper.

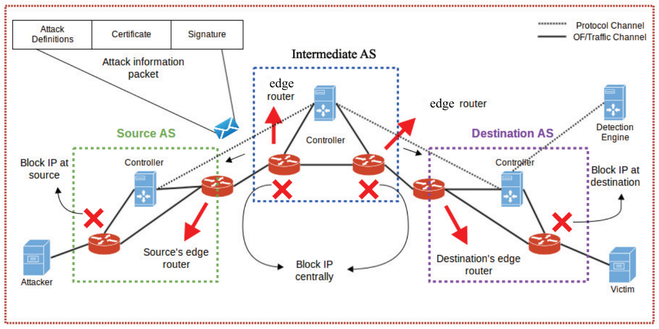


Fig. 1. High-level Architecture of Collaborative DDoS Framework

II. System Design and Architecture

Collaborative DDoS mitigation requires multiple SDN domains networked together as depicted in fig. 1. Each domain is a complete AS with egress and ingress routers. Any single AS may comprise of multiple SDN controllers which communicate with each other via our proposed C-to-C protocol. At the border of any AS sits SDN controllers that are capable of communicating with the neighboring AS's controllers to transfer attack definitions[2]. These ASs can roughly be divided into a *Source Domain*, an *Intermediate Network Domains* and a *Destination Domain*.

The source network is the one attack traffic is initiating from. The intermediate network(s) is comprised of multiple SDN domains connected with each other. The destination network(s) is the one victim is residing in. Attack traffic initializes from the source domain(s). It passes through intermediate networks to reach at the destination network.

In this paper, we have leveraged SDN to effectively mitigate the DDoS attack closest to the source. Our primary assumption in this work is that a *detection engine* will inform our SDN controller about possible attack information based on which we will mitigate the DDoS attack. This detection engine may consist of very effective and sophisticated detection mechanisms, like the one proposed in [12], which can be both internal as well as external to the AS. In the following subsections we have discussed the internal component architecture of the controller. Further, we have also elaborated the payload structure of the C-to-C protocol and summarized the overall collaborative DDoS mitigation work flow.

A. Controller-to-Controller (C-to-C) Protocol:

Figure. 2 depicts a typical packet sent from the detection engine to the SDN controller. A typical C-to-C packet sent by the detection engine to the SDN controller comprise of three sections i.e data, certificate and signature. Data section contains a list of IPs and the corresponding action needed to be taken. Certificate section contains a certificate along with the public key. Signature section contains a message digest signed with the private key of the attached certificate.

1) Data Section: This section contains all the information that needs to be communicated, which in our case are typically a list of IPs along with their statuses. Figure. 3 shows a raw representation of the data contained in JSON format. The JSON object is self-descriptive. We have a list of IPs that are needed to be blocked or if an IP was previously blocked mistakenly then the status helps in unblocking it.

[2]Attack definition, attack information and flow policy are used interchangeably.

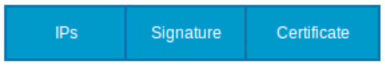
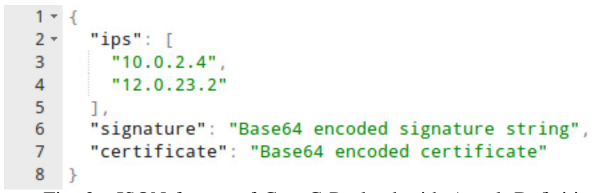


Fig. 2. Payload Structure of C-to-C Protocol



```
{
  "ips": [
    "10.0.2.4",
    "12.0.23.2"
  ],
  "signature": "Base64 encoded signature string",
  "certificate": "Base64 encoded certificate"
}
```

Fig. 3. JSON format of C-to-C Payload with Attack Definitions

2) Certificate Section: The certificate section comprise of a certificate attached by the communicating system to authenticate its legitimacy.

The idea of certificate chaining is not new and it is heavily used in day to day communications, like in authenticating the DNS records, in client to server communication and server to server communication. There are two types of certification authorities (CAs): root CA and intermediate CA. In order for a certificate to be trusted, it must have been issued by a CA that is included in the trusted store. In our system a trusted store is a directory containing root or intermediate certificates and other private keys of the user. There is no particular directory specified in Linux for trusted store. We have created our own in the POX controller folder.

If a certificate presented by a neighboring controller is not issued by a trusted CA then the certificate of the issuing CA is checked to see if the certificate of the issuing CA was issued by a trusted CA and so on until either a trusted CA is found (at which point signature is verified and flows are installed) or no trusted CA can be found (at which point whole payload is dismissed).

3) Signature Section: This section contains a message digest signed by the private key of the certificate attached. This helps in verifying the authenticity as well as integrity of the attack definition and its sender.

B. Controller Modules

We have written different programs that run on POX [3] as stand-alone modules. These modules allow the controller to perform different functionalities such as installing flows, listening for attack definitions from neighbouring controllers, validating the signature of attack definitions and propagating the attack definitions to other controllers (see fig. 4). The modules are further discussed in the following sections.

1) Policy Listener Module: This module runs a simple lightweight server program on the controller that listens on a predefined port for attack definitions received from neighboring controllers. On receiving an attack definition the module verifies the payload via Payload Validation Module using the embedded certificate. Upon the successful verification all the attack definitions are written on a CSV file and the l3 Learning Module is made aware of the updated policies. The l3 Learning

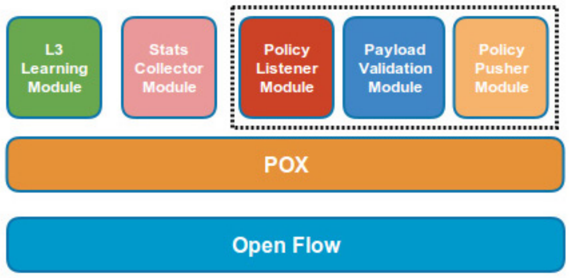


Fig. 4. Component Architecture of Controller

module then refreshes the policies by installing new flows from the updated CSV file. This module also calls the Policy Pusher Module to forward the flows to the neighboring controller.

2) Payload Validation Module: This module validates the certificate and verifies the signature of the payload before it is further processed and flows are installed into the individual nodes (i.e. switches or routers). The certificate is validated via chain of trust. A root certificate of the CA is present in the trusted store. Certificate is validated against the trusted CA. Upon the successful validation of the certificate the signature of the payload is validated for checking the integrity of the message. The IPs contained in the payload are forwarded to the connected nodes upon successful signature verification.

3) Policy Pusher Module: This module pushes the policies (containing the new attack definitions) to the neighboring controllers. The Policy Listener module informs the Policy Pusher module to update the policies locally upon the successful verification and forward the attack definitions.

4) L3 Learning Module: This module derives most of its functionality from POX's out of the box forwarding module named l3_learning. It is a simple layer 3 learning module that provides connectivity between the nodes via the nodes they are connected with. Along with the connectivity, it installs policies received to block the attack traffic. When ever new flows are installed the policy listener module informs the l3_learning module. The l3_learning module then flushes all the flows installed on the nodes and install negative flows blocking malicious traffic.

5) Stats collector Module: This module collects information like number of packets/second passing through a particular domain, active flows installed in a network and traffic passing in Mbps, .etc. This module is specifically used to collect evaluations and results when the proposed mechanism is deployed on several testbeds.

C. Work-flow of Inter AS Collaborative DDoS Mitigation

The complete work flow of the Collaborative DDoS mitigation is summarize as follows.

1) The detection engine communicates with the SDN controller via C-to-C protocol and forward a list of malicious IPs in the form of an attack definition.
2) The SDN controller first validates the communicating server by going through the following steps:
    a) A certificate is retrieved from the payload.
    b) The Payload validation module validate the certificate via a root certificate of the issuing CA present in the trusted store.
    c) Once the certificate is authenticated via root chaining, the signature of the message is validated.
    d) Upon the successful validation of the signature the payload is further processed, otherwise it is discarded.
3) The IPs present in the payload are then written to a policy file and L3-Learning module is informed about the updates in the policies.
4) The L3-Learning module then reads the updated policies from the policy file
5) The L3-Learning module then installs the new policies on each connected node.
6) As a result of the new policies, malicious flows are blocked. Any previously blocked flows can be allowed depending upon the improvised detection.
7) The SDN controller then forwards the policies to the neighboring controllers via Policy Pusher Module.
8) The neighboring SDN controllers performs the same steps starting from step 2 to 7.

D. Protecting Controllers Against DDoS

C-to-C protocol facilitates selective communication between authorize controllers with valid signatures on the payload. This enables effective filtering of unnecessary traffic from unknown source(s). Attackers can try to compromise a controller to launch a DDoS attack on the neighboring domains, however, compromising a legitimate controller in an ISP (or AS) is synonymous to compromising the ISP itself. Discussion on such attacks and its defense is beyond the scope of this paper.

III. TESTBED AND EVALUATIONS

We distribute our testbed in three different networks i.e. *Source, Intermediate* and *Destination* network(s). We use Mininet [2] to emulate the networks with POX [3] as the controller platform. In our testbed, OF-switch are also used to simulate the behavior of an edge router in an SDN network to filter the traffic as per policy. All the Mininet instances emulating different networks are connected via GRE tunneling. The role of each network in our testbed is discussed below.

The *Source* network is the one that generates both legitimate and attack traffic. We used three nodes in the source network out of which two generate attack traffic while one node is the legitimate one.

The *Intermediate* or inter-connecting networks are multiple Mininet networks connected via GRE tunneling. They are autonomous networks running their own topologies and also act as the transit networks to route the traffic between source and destination. They can be treated as different autonomous systems within the same ISP or different ASs in different ISPs. Since they are Mininet emulated networks they contain SDN controllers running on POX framework.

The *Destination* network is also a Mininet network comprising a victim node that is the destination for both legitimate and

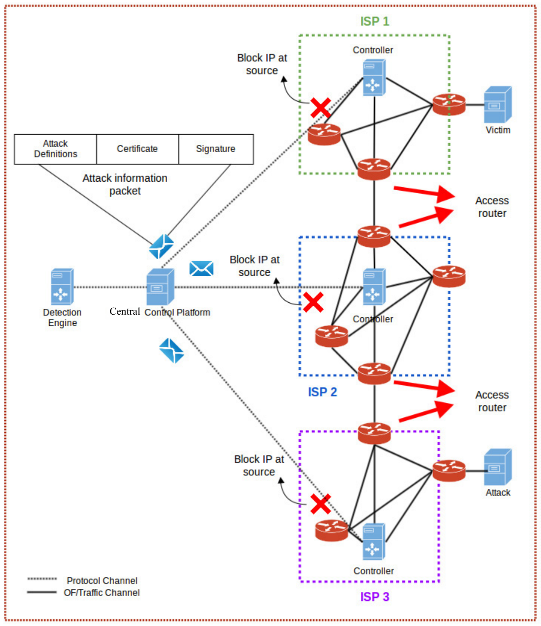


Fig. 5. Centralized Policy Distribution

attack traffic generating from the source network(s). Initially the destination host fulfills all the requests coming from the source network(s) without any distinction of the legitimate and malicious traffic. But once the destination network is made aware of the malicious traffic, it starts blocking the malicious traffic and subsequently inform the neighboring networks.

For most part of our evaluations (mentioned otherwise) each node in our testbed consists of 2.60 GHz Intel core i5 CPU, 8 GB RAM, 500 GB HDD and 1 Gbps Ethernet card. We use Scapy [4] to generate ICMP packets with varying payloads for both attack and legitimate traffic targeted towards the destination.

Since detection is not within the scope of our work, we have simulated a node as a detection engine that feeds malicious flows to the destination network in order to mitigate the attack. It can reside in any of the above networks or it can be in different network. Further there is no restriction that this node should also be running within an SDN network. It can be in a legacy network. All that is required from this node is to speak the same C-to-C protocol as defined in the above section to properly authenticate itself and provide the attack definitions.

A. Deployment Approaches

We used three different approaches to deploy our testbed. The basic difference between the approaches is how the policies (attack definitions) are distributed. These approaches are briefly discussed as follows.

1) Linear Approach: This is the regular implementation discussed in the above section with architectural details (fig 1). This approach comprised of all the participating networks i.e. *Source, Intermediate* and *Destination* networks connected with each other in linear fashion. A third party detection engine (similar to HADEC [12]) feeds the attack definitions into the destination network, which then forwards them to the neighbouring network and this process continues till the definitions reaches the source.

2) Centralized Approach: In this approach (see fig. 5) there is one *Central Control Platform* that handles attack definitions from all the connected networks. Upon successful verification the platform then forwards the attack information to the connected SDN controllers. The SDN controllers upon receiving the attack definitions install the flows defined in the policy after verification check. This approach is helpful in preventing hop-by-hop dissemination of attack definition, specially in scenarios where a huge amount of traffic is being handled by SDN controllers. Moreover in this approach the flows installed can be targeted depending upon the destination address. As a result not every SDN network has to install all the flows. The central approach only forwards the relevant flows to the intended SDN controller hence saving the TCAM memory of OpenFlow switches.

3) Mesh Approach: In this approach any single network deploys a mesh connectivity with existing networks. This enables any single network to forward the received attack definitions directly to all the connected networks, instead of pushing the attack definitions linearly one by one to its neighboring network. This way the mitigation process is very fast.

B. Bootstrapping

In order to effectively bootstrap the proposed scheme, we have to consider the accessibility to neighboring controllers and pre-hand knowledge of the CA. In case of peer-to-peer deployment, a controller in an AS will follow the peering agreements. Just like any edge routers are configured with the accessibility information of neighboring AS's edge router, the controller's in peer ASs will be configured with the accessibility information (IP, port). In centralized approach, the AS can publish list of authorized controllers in the *Central Control Platform*. This approach is very simple, yet very effective and it is successfully being used by Sender Policy Framework (SPF) [5] (an IP based email authentication mechanism with over 7 million registered domains). The knowledge of CAs is part of controller's configuration, this approach is successfully used in DNSSEC and all the browsers have pre-installed certificates of more than 600 root CAs and 1200 intermediate CAs.

C. Effect of Deployment Approaches on Attack Mitigation

We performed different experiments to analyze the behavior of attack mitigation under different deployment approaches (linear, central and mesh). We would like to emphasize that the core focus of the proposed collaborative scheme is mitigation of DDoS along the attack path and this scheme can be flexibly appended with any effective detection algorithms. We got some promising results that give us insight about the potential problems that our proposed architecture is capable of solving.

1) Linear: For Linear approach we setup a testbed with eight networks (one source, one destination and six intermediate networks) connected in a linear fashion. The choice of

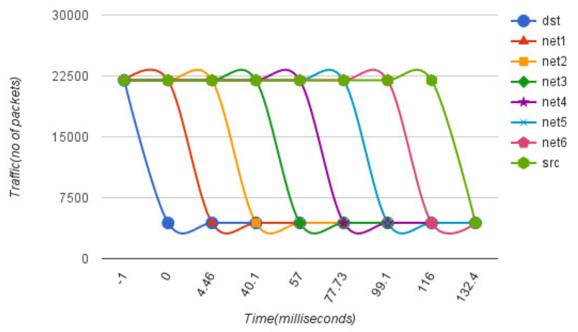


Fig. 6. Mitigation Effect with Six Intermediate Networks in LAN Setting

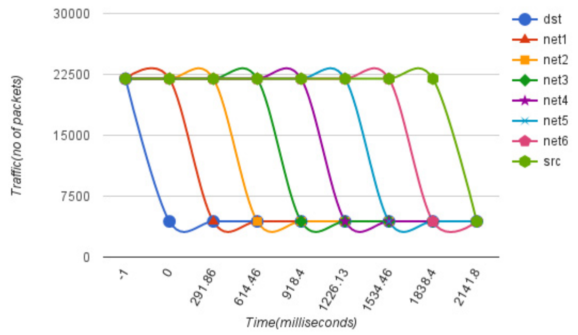


Fig. 7. Mitigation Effect with Six Intermediate Networks in ISP Setting with AS-AS Latency and Processing Delays

six intermediate networks give us relevant ISP settings that would work in practice. This is because the average length of AS paths over time, as seen by the RIPE NCC Routing Information Service (RIS) route collectors, for IPv4 networks is fairly stable at 4.3 hops [6].

In the experiments, the source network generates approx. 21,960 packets per second out of which only 4,392 packets are legit and the rest of 17,568 packets are malicious. For the sake of simplicity we have assumed that none of the intermediate network is generating its own traffic. Hence the only traffic passing through the intermediate networks is coming from the source network.

For the first experiment, we used a LAN settings with no delays between different SDN domains. Further we used attack definition with 1K IPs to keep the processing delay minimum. The results generated via this setup are shown in fig. 6. This graph is between the data flowing from source to destination and the time it takes to mitigate the attack.

At time -1 the attack is being carried out, so the amount of traffic in all eight networks is at the maximum volume. At time zero, our destination network receives the attack definitions via detector node. The SDN controller at the destination node verifies the authenticity of the attack definitions from the detector node and upon success installs the flows. Due to this, we observed a traffic drop at destination network and the number of accepted packets are reduced to only the legitimate ones only i.e. 4,392. At this moment the amount of packets flowing through other networks remains the same. After installing the flows in its own network the SDN controller at the destination network forwards the attack definitions to the neighboring network i.e network 1. Network 1 validates the source of the message and installs the flows. Due to which at time 4.5 ms there is a decrease in the traffic at network 1. The network 1 follows the same pattern and forwards the attack definitions to its neighbor i.e. network 2. Network 2 follows the same steps too and at time 40 ms the traffic flow drops to normal only allowing the legitimate traffic to pass through. This goes on till network 6 forwards the attack definition to the source network. At time 132 ms the source network installs the flows and the traffic drops to the legitimate traffic only. In the end, the attack has been mitigated not only from the destination network, but all the way to the source with the help of collaborative propagation of the attack definitions. Here, we also observed that the validation and processing of small size attack definitions has trivial impact on the latency.

In our second experiment we focus on the real world deployment aspect of an ISP settings. We added AS-to-AS communication latency and processing delays for large size attack definitions. For AS-to-AS communication latency we ran traceroute for arbitrary domains and took worst case estimates of 150 ms avg. delays using a 500 Kbps Internet connection (to simulate low available bandwidth during an ongoing DDoS). It took on avg. 137 ms to process a payload containing 100K IPs (see §III-D). Here, just like the previous setup we have one source network, one destination network and six intermediate networks. The whole operating procedure remains the same as thoroughly described above. The results shown in fig. 7 resemble the pattern of fig. 6 except for the values. The effect of mitigation is instantly transferred from destination to the source. One thing worth noting is the amount to time it takes to mitigate the attack completely all the way from the destination to the source is approx. just 2141 ms or 2.14 seconds.

Larger number of Intermediate networks have proportional increase in the mitigation time. Nevertheless, our proposed framework and C-to-C protocol is lightweight with instantaneous effect. It only requires somewhere between 290 to 330 ms to process and forward attack definitions from one network to another.

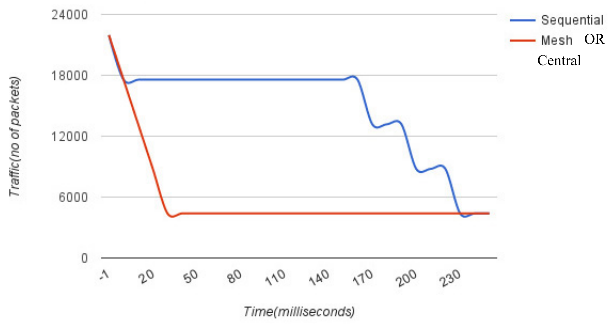


Fig. 8. Linear vs Mesh or Central: Comparison of Policy Propagation time

2) Mesh and Centralized Approach: In fully meshed and centralized approach the controllers are directly connected with each other. This way the attack definitions or flows are pushed from the destination to Intermediate and source networks. We performed similar experiment as discussed above, but with mesh of controllers connected with each other or centralized way of carrying out communication the effect is

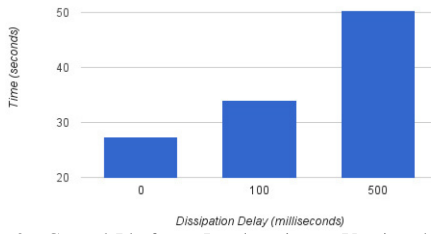


Fig. 9. Central Platform: Load testing at Varying delays

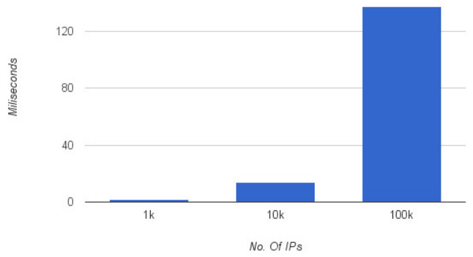


Fig. 10. Payload Processing Delays

very immediate as compared to the linear approach. Figure 8 shows immediate drop in attack traffic since the controllers are connected directly and the flows are pushed right from the destination or central platform to the individual networks.

D. Performance of Central Control Platform

The main idea of central control platform is to create a trusted authority that verifies the attack definitions received so that intermediate controllers do not have to go through the laborious task of individually verifying and forwarding the attack definitions. The result of this approach is quite similar to the one achieved in Mesh approach. In this section we evaluate the dissemination delays of the flows and the effect of payload size on the central controller. For the performance evaluations we used a low end machine with Intel Core i3-4010U CPU @ 1.70GHz 4 with 4.0 GB of RAM as the central controller.

In order to evaluate the dissemination delays of attack definitions, we generated 100 different payloads with attack definitions and forwarded each payload to 100 connected controllers in three different modes. i.e. Burst mode, with 100 ms delay (the delay added between two different attack definitions), and with 500ms delay. Figure. 9 shows the results. In burst mode i.e. with 0 delay between attack definitions, it took approx. 28 seconds to dissipate all the flow policy to 100 connected controllers. With a delay of 100ms it took approx. 34 seconds and with a delay of 500 ms it took approximately 50 seconds.

The performance of our system also depends upon the payload size of an attack definition which mainly consists of malicious IP addresses. We took various payloads and computed the time to verify and process the IPs to generate relevant flow table entries. Figure. 10 display the effect of increasing size of payloads. It took on average 1.8 ms to process (verification of signature and insertion of flow table entry) a payload containing 1K of IPs. The processing time increases to 13 ms for payload with 10K IPs and around 137 ms to process 100K IPs.

IV. Conclusion

In this paper, we present a lightweight, efficient and easy to deploy collaborative DDoS mitigation scheme leveraging SDN. Using the proposed scheme a SDN controller in any AS can directly communicate with the controllers in the adjacent network via secure C-to-C protocol and inform them about an ongoing attack. This helps in efficient propagation of attack definitions all the way from the victim to the attack sources. We also introduced three different deployment approaches i.e. linear, central and mesh in our testbed and tested the overall efficiency. The evaluation results showed that the effect of mitigation is instantaneously transferred from destination to source. It took around 2.14 seconds to mitigate the attack in an eight hop linear deployment. Further, it only requires somewhere between 290 to 330 ms to process and forward attack definitions between adjacent networks. The processing of attack definition payload (verification of signature and insertion of flow table entry) is also lightweight even on low end machines with a processing time of around 13 ms for a payload with 10K IPs.

References

[1] Dyn cyberattack. www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet.
[2] Mininet. www.mininet.org/.
[3] Pox controller. www.github.com/noxrepo/pox.
[4] Scapy. www.secdev.org/projects/scapy/.
[5] Sender policy framework. http://www.openspf.org/.
[6] Autonomous system path lengths. https://labs.ripe.net/Members/mirjam/update-on-as-path-lengths-over-time, 2012.
[7] M. Belyaev and S. Gaivoronski. Towards load balancing in sdn-networks during ddos-attacks. In *Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), 2014 International*, 2014.
[8] R. Braga, E. Mota, and A. Passito. Lightweight ddos flooding attack detection using nox/openflow. In *35th IEEE Conference on Local Computer Networks (LCN)*, 2010.
[9] Nhu-Ngoc Dao, Junho Park, Minho Park, and Sungrae Cho. A feasible method to combat against ddos attack in sdn network. In *International Conference on Information Networking (ICOIN)*, 2015.
[10] K. Giotis, G. Androulidakis, and V. Maglaris. Leveraging sdn for efficient anomaly detection and mitigation on legacy networks. In *Third European Workshop on Software Defined Networks*, 2014.
[11] K. Giotis, M. Apostolaki, and V. Maglaris. A reputation-based collaborative schema for the mitigation of distributed attacks in sdn domains. In *IEEE/IFIP Network Operations and Management Symposium*, 2016.
[12] S. Hameed and U. Ali. Efficacy of live ddos detection with hadoop. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, April 2016.
[13] J. Jeong, J. Seo, G. Cho, H. Kim, and J. S. Park. A framework for security services based on software-defined networking. In *Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on*, 2015.
[14] S. Lim, J. Ha, H. Kim, Y. Kim, and S. Yang. A sdn-oriented ddos blocking scheme for botnet-based attacks. In *Sixth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2014.
[15] Q. Yan and F. R. Yu. Distributed denial of service attacks in software-defined networking with cloud computing. *IEEE Communications Magazine*, April 2015.
[16] Xiaowei Yang, David Wetherall, and Thomas Anderson. A dos-limiting network architecture. In *ACM SIGCOMM Computer Communication Review*, volume 35. ACM, 2005.
[17] Xiaowei Yang, David Wetherall, and Thomas Anderson. Tva: a dos-limiting network architecture. *IEEE Transactions on Networking*, 2008.