Detection of DNS based DDoS attacks in an SDN environment using Machine Learning techniques

Vinayak Gupta[a] , Harjas Singh Bathla[b]

[a] Undergraduate, Dept. of Computer Science & Information Systems, Birla Institute of Technology & Science (BITS), Pilani, Vidya Vihar, Rajasthan, 333031, India

[b] Undergraduate, Dept. of Computer Science & Information Systems, Birla Institute of Technology & Science (BITS), Pilani, Vidya Vihar, Rajasthan, 333031, India

Abstract- Distributed Denial of Service (DDoS) attacks are one of the biggest concerns in network security as they attempt to hinder a victim's ability to access service by either exploiting some protocol bug or by targeting the victim's bandwidth. Because the attacker can forge packets using IP spoofing, the source of the attack is very difficult to identify. We present a system in this paper that is cable of detecting a DNS based DDoS attack. Using Software Defined Networking (SDN), the switches can be configured in such a way so as to collect only DNS based traffic and then by employing machine learning techniques, one can detect a DDoS attack is happening or not. We make use of a K-means++ clustering algorithm to detect the attack and we also present a novel way to conduct a DDoS attack in an SDN environment using Mininet and POX controller.

I. INTRODUCTION

Today's network being unchanging and oriented towards a specific service is prone to Distributed Denial of Service (DDoS) attacks. In DDoS, the attackers try to get control of various bots which send large amounts of packets to the victim exhausting its bandwidth and other resources. Two major kinds of DDoS attacks have come to surface in the recent years [1]. The first one is protocol exploitation flooding attack in which the attackers use the flaws and bugs present in the victim's protocol to exhaust the victim's resources e.g. TCP SYN flood. The other kind of attack is DNS amplification which directly targets the victim's bandwidth. The latter attack is the focus of this paper as it allows the attacker to conduct an attack on a larger scale because it can create a disproportionate amount of traffic at the victim. Moreover, using IP spoofing and reflectors, the attacker can hide his identity.

In a DNS amplification attack, the attacker sends a forged DNS query packet with the source IP of the victim to the DNS server. As the DNS service works on UDP connection protocol, the authenticity of the source is not required, and the server reflects the forged packet to the victim. Since the size of the response is much larger than the query packet, this kind of attack is called an amplification attack. This response packet is received by the victim which processes it and later discards it. This exhausts not only the victim's bandwidth but also increases the

computational load on the DNS servers. This attack can be conducted on a large scale given there are multiple reflectors under the control of an attacker and provided they send the DNS query packets with the same spoofed IP address. DNS amplification attacks have been used in past on various organizations. In March 2013, an amplification attack was orchestrated on Spamhaus [2], a spam email tracker which brought down the Internet. Moreover, the attack was so anonymous that no one has yet been able to find its source.

SDN is a technology that aims at making the networks more agile and flexible. It enables service providers and enterprises to integrate various kinds of services to the same infrastructure. In an SDN environment, a centralized controller is present which can be used to configure the switches without actually touching them. This is possible because of the decoupling of the data plane from the control plane [3]. The data plane represents the underlying switches and the control plane is the centralized SDN controller which makes the traffic decisions for every switch.

This paper proposes a solution to detect DNS based amplification attacks using a POX controller in software-defined networking (SDN) environment. An SDN barrier (a collection of switches) is designed at the network level in between the autonomous systems (AS) and ISPs (Internet Service Providers) and all the traffic passing through the barrier is collected with the help of a collecting agent commanded by a controller. Since we are working in an SDN infrastructure, we can configure the collecting agent in such a way that it collects traffic corresponding to DNS only i.e. traffic directed to or coming from port 53. While the traffic is collected after every n-secs (n = 60 in our simulation), a detecting agent works in the background to cluster the traffic statistics using the K-means++ algorithm. As soon as an anomalous cluster is detected, the detecting agent would inform the network manager who can then address the incoming traffic.

This paper is further divided into the following sections: In Section II, we present the related work that has been done in this field and how our method differs from them. Section III presents an overview of the K-means++ clustering method and an explanation is provided for why it is used over the other clustering algorithms. Section IV introduces the formal solution to the problem and Section V describes the experimental setup. All the results are provided in Section VI and finally, we conclude in Section VII.

II. RELATED WORK

DDoS attack nowadays has become a major cyber-attack. So, researchers have analyzed and proposed many methods by surveying large scale internet events. Since our work deals with the detection of DNS based DDoS attack using machine learning techniques, we mention some related work on this proposed method. Most of the related work has been done in the recent decade and classification algorithms have been used widely for the purpose. Rastegari et al. [4] use neural networks as well as support vector machines (SVM) to detect the attack in a simulated environment. Also, Meitei et al. [6] use DNS request and response packet data to train their dataset. Various classification algorithms like SVM, Decision Tree, Multi-Layer Perceptron and Naïve Bayes have been used in their paper to classify the traffic into DDoS attack and normal

traffic. Chen et al. [1] present a way of preventing DDoS attacks in an SDN environment while also detecting the attack using SVM classifier. In Muragaa et al. [2], web traffic is collected using Mininet and POX controller and presents an interesting overview of SDN.

III. OVERVIEW OF K-MEANS++ CLUSTERING

K-means++ clustering is an initialization process of the well-known K-means clustering algorithm. K-means is an unsupervised machine learning algorithm which means that it does not require a training set to begins its clustering process. In K-means, K represents the number of clusters into which the whole dataset needs to be classified and what makes it so useful is that it is fast and efficient and works very well on small as well as large datasets. However, the only drawback of K-means clustering is that since it initializes the centers of the clusters in a random way, it might give very poor clusters. This is the benefit of K-means++ algorithm where the centers are initialized in such a way that the intra-class variance is minimized where variance is defined as the sum of squared distances of every point from the center of the cluster.

K-means++ clustering algorithm starts with the user choosing the value of K. This can be determined by the Elbow method which looks at how much variance is explained as a function of the number of clusters. The K-means++ algorithm then selects the appropriate centers of the clusters. Every data point is then assigned to the closest cluster and the center of the cluster is recomputed. Thus, every cluster gets a new center, and this continues until the position of center does not change for every cluster. At the end of the clustering process, all the data points are classified into the K clusters defined initially.

IV. SYSTEM ARCHITECTURE

The solution that we propose is used to prevent the bandwidth of the links in the autonomous system from getting exhausted. Hence, it is implemented on the network side instead of the victim's side. The whole solution can be effectively divided into two parts: a collection agent and a detection agent. A group of switches which we call as an SDN barrier is used to collect all the traffic into or from the autonomous system (AS). This is implemented between an AS and ISP and hence, can be used by many AS and many ISPs as shown in Fig.1. The SDN barrier is configured using a POX OpenFlow controller which instructs the switches to collect only the DNS based traffic. Because most of the modules of the POX controller are written in Python, we choose Python as our scripting language. A code runs on the kernel of the controller and it guides the switches in the SDN barrier to maintain a log of various attributes of forward and backward packets passing through it. The following attributes are collected:

- Forward Packet count
- Backward Packet count
- Backward Packet size

These three attributes are enough to detect a DDoS attack as when an attack occurs, the forward packet requests are not able to receive their associated responses, hence the backward packet

count decreases. Also, the size of individual backward packets is quite large when a full DDoS attack occurs as unrequested packets are received on the victim's side by the compromised DNS server.

By design of the OpenFlow switches, the flow tables contain flow entries which describe the path that should be taken by the packets when arriving at the switch [3]. These flow entries are added according to the instructions by the controller. Moreover, the switches maintain counters which store the number of packets and bytes passing through it. By checking the source and destination port of every packet passing through it, the packets can be classified as backward or forward. If the source port is 53, the packet is considered as backward while if the destination port is 53, the packet is described as forward. The Python script runs on the controller and it collects these statistics from the counters after every 60 seconds. All the statistics collected are exported to a "csv" file where all the features are treated as columns and the values collected every 60 seconds correspond to the rows. The csv file is then passed onto the detection agent.

The detection agent is a Python module that uses the K-means++ clustering algorithm to categorize the networks statistics that the collection agent passes to it. Since the algorithm is unsupervised, no training of data needs to be done. The optimal number of clusters is determined by the Elbow method and as soon as an anomalous cluster is encountered, the agent informs the network manager of the same. Usually, the anomalous cluster has a large backward packet count as compared to the forward packet count. This happens because the forward packet count represents the legitimate traffic that all the AS are sending. However, when a DDoS attack happens, some unrequested packets are also received by the AS. The backward packet count comprises of the legitimate replies as well as the DDoS traffic due to IP spoofing of a victim in the AS. The detection agent module runs every x seconds which determines the frequency of detecting the DDoS attack. Once a DDoS attack is detected, the network manager can block all the traffic coming from the attacker into the switch in the SDN barrier that is in the path of the victim's AS. Bloom filters can be employed by the manager to drop the attack traffic.

V. EXPERIMENT SETUP

We have used Mininet [7] for the simulation of the experiments. Mininet is a network emulator which runs a collection of end-hosts, switches, routers, and links on a single Linux kernel. It uses lightweight virtualization to make a single system look like a complete network, running the same kernel, system, and user code. VirtualBox [8] is used for running the Mininet virtual machine and POX controller, which is a python based SDN controller platform, is used for creating the learning switch.

To simulate the DDoS attack, the topology shown in Figure 1 was created:

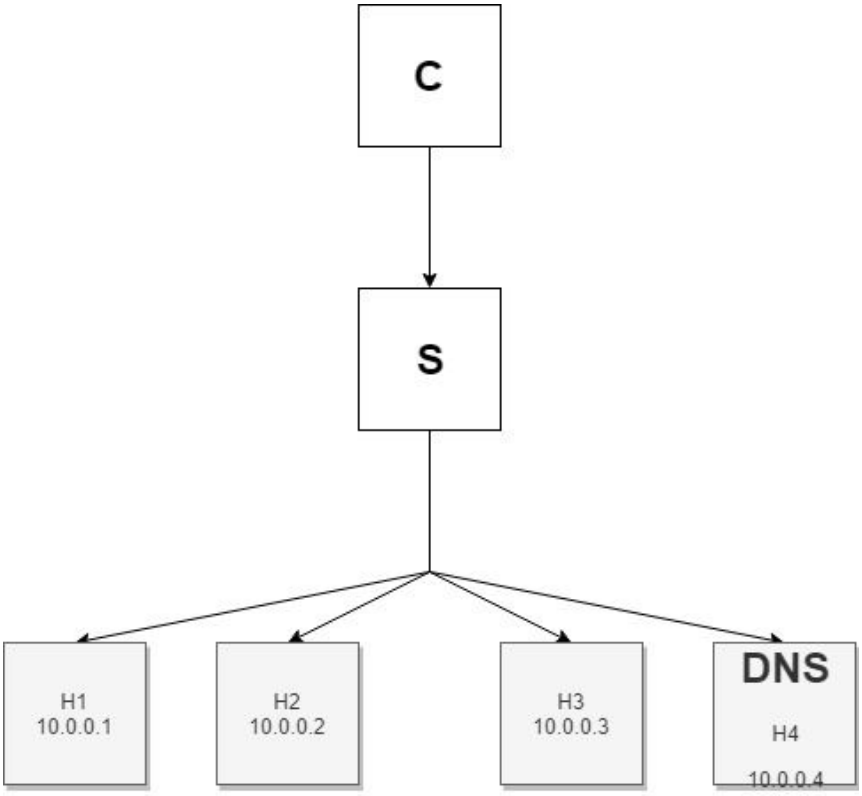

Figure 1. The topology used in the experiment

- C depicts the controller and S is the OpenFlow switch.
- H4 is a custom DNS server. It has two zone files - one for a query having a normal size response and one for the DDoS attack having an abnormally large size.
- H1 is the target host for the DDoS attack. Also, H1 will query the DNS server for generating normal traffic.
- H2 and H3 are the hosts which will launch the attack.

Scapy[9], a python library was used for generating the DNS queries. It allows the user to customize various packets by modifying any field in the packet header, such as IP address, MAC address, etc. and send them out with the Scapy API.

To simulate a reflection attack, the packets sent by the attacking hosts had a forged source IP address, that being of the target host H1 in our case. In addition to this, a separate script runs on the switch during the experiment. The purpose of this script is to collect the data statistics (as mentioned in Section V) passing through it. This data is then used for detecting a potential DDoS attack through the machine learning model. The data is collected for different frequencies of packets sent and each such attack is simulated for 5 minutes on the above-mentioned topology.

Table 1. Description of parameters

Parameter	Explanation
forward_packet_count	Number of DNS packets arriving at the switch with the destination port of 53
backward_packet_count	Number of DNS packets arriving at the switch with the source port of 53
backward_flow_volume	Amount (in Bytes) of DNS packets arriving at the switch with the source port of 53

Table 2. Description of the data(.csv) files mentioned in the Appendix (Section IX)

File Name	Number of Attacker Hosts	Frequency of Packets Sent
sdn_attack_1_30	1	30
sdn_attack_1_60	1	60
sdn_attack_1_90	1	90
sdn_attack_2_30	2	30
sdn_attack_2_60	2	60
sdn_attack_2_90	2	90
sdn_normal	-	-

We gathered DNS flows for normal traffic using Wireshark [10] and created sample vectors from them. Along with the l2_learning script and the data collection script, a machine learning script also runs simultaneously on the switch. It takes as input the data collected from the data collection script after every 1 minute (as mentioned in Section V) and clusters them in order to detect a possible DDoS attack.

VI. RESULTS AND INFERENCES

In this section, we present the results of the K-means++ clustering algorithm on the collected network statistics. By using the Elbow method as defined in Section IV, the optimal number of clusters are determined which come out to be 3. Table 3 presents the inferences that can be derived from each of the clusters.

Table 3. Clustering Inferences

Conditions	Inference
Forward packet count is approximately equal to the Backward packet count	Normal Traffic
Responses are not being received for each forward packet, hence the forward packet count is much larger than the backward packet count	Intermediate DDoS attack
Large and unrequested backward packets are being received	Full DDoS attack

When the normal traffic is coming through the SDN barrier, the forward packet count is approximately equivalent to the backward packet count because each request receives a corresponding response. However, when an attacker starts conducting a DDoS attack, the bandwidth of the link between the SDN barrier and the victim host starts getting utilized and not every forward packet request receives a corresponding response. Hence, the forward packet count exceeds the backward packet count by a large amount. When the DDoS attack is working in its full strength, unrequested packets are received by the victim host. The victim receives packets with abnormally large sizes from the DNS server. Figure 2 shows the results of the clustering in graphical form on the files of sdn_attack_1_90 and sdn_attack_2_60 where backward packet ration represents backward flow volume divided by the backward packet count. The values have been scaled using the StandardScaler module of the Scikit-learn class of Python.

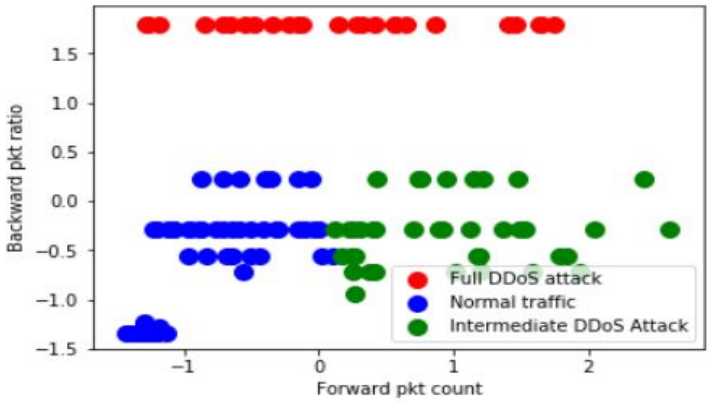

Figure 2(a). Clustering results on file sdn_attack_1_90

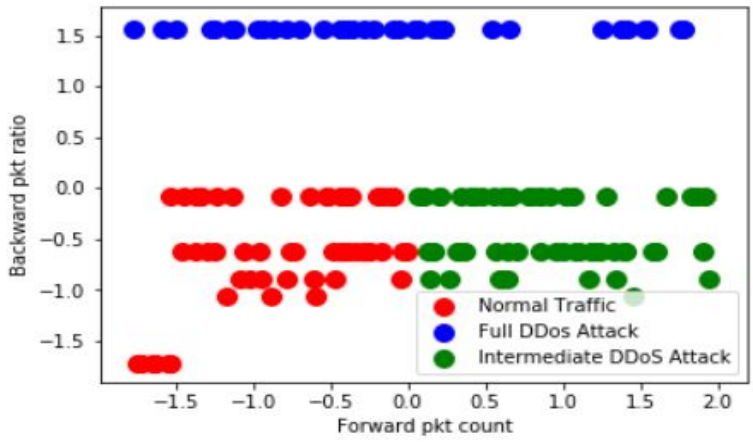

Figure 2(b). Clustering results on file sdn_attack_2_60

VII. CONCLUSION

In this paper, we have presented a novel way of detecting DDoS attack on the network side where the whole AS can be prevented from the attack using an SDN barrier. The SDN environment allows us to consider only DNS based packets which drastically reduces the load on the collection agent. A DDoS attack has been conducted in a simulated environment and promising results have been obtained while clustering the statistics. The algorithm does not require prior knowledge of whether a DDoS attack is happening or not to detect an attack.

VIII. APPENDIX

The detailed steps for setting up and simulating the experiment for conducting the DDoS attack can be found at https://github.com/harjas27/DNS-based-DDoS-in-SDN/

The repository also contains all the data files and python scripts used in the experiment.

IX. REFERENCES

[1] C. Chen, Y. Chen, W. Lu, S. Tsai and M. Yang, "Detecting amplification attacks with Software Defined Networking," *2017 IEEE Conference on Dependable and Secure Computing*, Taipei, 2017, pp. 195-201.doi: 10.1109/DESEC.2017.8073807

[2] C. MacFarland, Douglas & Shue, Craig & J. Kalafut, Andrew. (2015). Characterizing Optimal DNS Amplification Attacks and Effective Mitigation. 8995. 15-27. 10.1007/978-3-319-15509-8_2.

[3] Muragaa, W.H., Seman, K., & Marhusin, M.F. (2017). A POX Controller Module to Collect Web Traffic Statistics in SDN Environment.

[4] S. Rastegari, M. Saripan, and M. Rasid, “Defending denial of service attacks against domain name system with machine learning techniques,” IAENG International Journal of Computer Science, 2010.

[5] D. Huistra, “Detecting reflection attacks in DNS flows,” in 19th Twente Student Conference on IT, 2013.

[6] Irom Lalit Meitei, Khundrakpam Johnson Singh, and Tanmay De. 2016. Detection of DDoS DNS Amplification Attack Using Classification Algorithm. In Proceedings of the International Conference on Informatics and Analytics (ICIA-16). ACM, New York, NY, USA, Article 81, 6 pages. DOI: https://doi.org/10.1145/2980258.2980431

[7] “Mininet,” https://mininet.org/

[8] “VirtualBox,” https://www.virtualbox.org/

[9] “Scapy project,” http://www.secdev.org/projects/scapy/

[10] “Wireshark,” https://www.wireshark.org/