Mitigation and Detection of DDoS Attacks in Software Defined Networks

Conference Paper · August 2018

DOI: 10.1109/IC3.2018.8530514

Krishna Asawa
Jaypee Institute of Information Technology
52 PUBLICATIONS **289** CITATIONS
SEE PROFILE

Proceedings of 2018 Eleventh International Conference on Contemporary Computing (IC3), 2-4 August, 2018, Noida, India

Mitigation and Detection of DDoS Attacks in Software Defined Networks

Shariq Murtuza
Dept. of CSE
Jaypee Institute Of Information Technology
Noida, India
shariq.murtuza@jiit.ac.in

Krishna Asawa
Dept. of CSE
Jaypee Institute Of Information Technology
Noida, India
krishna.asawa@jiit.ac.in

Abstract—**The Software Defined Networking (SDN) paradigm is expected to heavily integrate into future networks. Enterprises have already started migrating their networks to SDNs. Billions of smart devices constituting the Internet of Things will be connected to these high speed networks and will be communicating over these networks. The ubiquity of these networks along with the user devices connected to them becomes of paramount importance for the end users. This work presents a SDN switch based module to detect a Denial Of Service attack on the network and its connected components. The module analyzes each packet that comes to the switch and allocates a fitness score to each packet. The packets are labeled as safe, risky or dangerous and then they are either allowed to pass, proxied via a buffering system, or dropped immediately respectively.**

Index Terms—**Software Defined Networks, Internet of Things, Denial of Service Attacks, Traffic Filtering.**

I. Introduction

Internet of Things has started taking over the consumer market. These smart devices constitute the Internet of Things (IoT), that have very different communication requirements as compared to traditional network devices[1]. Maintaining flexibility and agility in such a huge network is very difficult and requires a robust controlling measure, and an inherent intelligence in the network [2].

A SDN [3][4] provides the ability for programming the network remotely and securely, while providing an abstraction to hide the internal components and their functionality [5].

This work discusses the different Denial of Service attacks that are possible on Software Defined Networks, along with various methods to identify and detect such attacks, and finally the methods to mitigate these attacks. The most common attacks that can be mounted are physical layer [6] based attacks, passive eavesdropping or active attacks on the transmitted data [7]. The state of art attacks on Software Defined Networks exploit the Data plane forwarding mechanism to flood the Control Plane using the Data Plane [8]. All these attacks exploit the inherent SDN behavior of forwarding the unseen packets to the controller via the control plane. This work presents a Decision Tree based method for identifying and mitigating DDoS attacks.

II. Related Work

A. State of Art DDoS attacks and their mitigation in SDNs

In Avant Guard [8] the authors have presented an attack on the control plane of a SDN by exploiting the forwarding nature of SDN switches. The attackers send a flood of new, unique and previously unseen packet connections. Each packet being unique, results in the switch forwarding to the controller. The attacker floods the data plane with such packets which results in the control plane being flooded with the query and decision packets. As a result the controller crashes immediately, or eventually the network link between the controller and switch crashes. To prevent these attacks the authors propose a proxying mechanism to withhold such packets streams unless they complete the TCP handshake. An attacker who has spoofed the source IP address will not be able to complete the handshake, thus only the connections completing the TCP handshake with the proxy module are proxied to the controller, else these packets are timed out and discarded. This proxy module itself can be targeted by a denial of service attack if the attacker starts sending multiple streams of packets that all require to be proxied.

LineSwitch [9] discusses and builds upon the shortcomings of Avant Guards proxying mechanism. Lineswitch proposes a probabilistic proxying addition to Avant-guards initial TCP handshake packets proxying along with network traffic blacklisting at the switch thus protecting the control plane. This new approach has the benefit of avoiding choking upon a flurry of incoming TCP packets. As soon as a packet arrives the switch takes into account multiple parameters and finds a weighted value based on these parameters. If the calculated value is is less than a decided threshold then only that SYN packet is proxied else it is sent directly to the controller.

III. Attack Model

A. DOS attack scheme

For our experiments, a DDoS attack [10] [11] was simulated on a Mininet [12] network having a SDN based backbone. This attack was a complex attack variant targeting a SDN switch. The SDN switch was the target destination of multiple spoofed TCP packets each with unique IP addresses [13]. To initiate a TCP based communication the SDN switch is required to create a reliable connection that includes performing a handshake with the sender. The SDN switch upon encountering a packet for which it is having no predefined rule, forwards the packet to the controller which then sends the decision back to the querying switch. The attacker tries to consume all the resource of the switch and the controller by swamping them with such packets, resulting in the eventual crashing of the components.

B. *Attack Detection*

We propose a *dynamic and adaptive threshold* for designing the class boundaries. The class boundaries keep on shifting as per the network traffic behavior. If the conditions become too constrained the boundaries relax and vice versa. This behavior is required since the network is highly unpredictable and the traffic can be very versatile in such scenarios. The attack exploited incomplete TCP handshakes, that were initiated but never completed. Soon the target resource pool was exhausted due to multiple such connections being initiated and never being completed. Before the earlier established connections could timeout, newer handshake connections in exponential amount were initiated resulting in the total resource exhaustion.

Our solution takes into consideration the complete packet parameters such as source and destination IP address [14], the past history and behavior of the sender's communication, payload type and the stream rate without any prejudice. If the adversary spoof's his/her source IP even then the payload would lead to the detection of a standard DDoS attack [15]. The system is precautious and by default labels the connection packet as risky packet. Over time the sender gains the trust of the system by behaving properly, still if the sender suddenly turns malicious, the system will allow only a single connection request un-proxied following which the system realigns and starts proxying the packets. The time-out for a connection keeps on reducing with each new connection created by the same sender and eventually leads to the blocking of the sender by labeling it as dangerous class.

IV. Fitness/Goodness Score Calculation

For classifying packets into safe, risky and dangerous category, firstly packets are alloted fitness or goodness scores. Once these scores are finalized, the packets are labeled into their respective classes depending on the class label threshold. We mention below the parameters used for calculating the packet scores. Each packet starts with a zero score initially, the below mentioned steps are taken to update the score of a packet.

1) If the source IP address of a packet is not having a pre-entry in the flow table of the SDN Switch the score of the packet is incremented by 1.
2) If the host is known to the switch for more than two successful previous communications then the score of the packet is reduced by the value 2. If the sending host has misbehaved in the past then its score is incremented by 2
3) If the packet is using a potentially risky protocol such as ICMP/IGMP or UDP then its score is incremented by 2, however due to the necessity of ARP/RARP protocols their usage attracts a lower penalty of incrementing the score by 1.
4) A very slow connection and a very fast connection both are a potentially problematic since they keep resources occupied. Either case can be a possible attack. Thus in both the cases the score is increased by 2.

V. Estimation of Class Boundaries/Threshold

Two thresholds are defined, first one, the risk threshold and the second one called as the danger threshold. A packet whose score is less than the risk threshold is labeled as a safe packet. The packets whose score is greater than risk threshold but lessor than danger threshold were labeled as risky packets whereas all the packets whose score exceeded the danger threshold were labeled as dangerous packets.

1) Deciding the Initial Threshold: A total of 10 iterations of the experiment were carried out initially to find appropriate thresholds, and the packets having a score higher than the threshold can be assumed as malicious. The packets from the KDD dataset [16] were sent to the classifier, which allotted each one a score. It was found that a large majority of packets having their score value less than *Four (4)* were safe packets, packet having score values greater than *Ten (10)* were dangerous attack packets. The packets having a score value in between *Four to Ten* them were in some cases malicious and otherwise legitimate, hence they were labeled as risky packets. To update the threshold value for each class a simple method is applied. To find the next iteration's boundary, we take the Harmonic Mean of the current iteration's threshold value for a particular class along with packet score of the packets that were mislabeled into the next immediate class. For example to find the new threshold between the safe and risky class, we take the harmonic mean of the current threshold along with the packet scores of the packets that were mislabeled as risky but were benign. Harmonic mean is taken as an arithmetic function since it is less susceptible to outliers

2) Proxying Risky Packets: As soon as a packet was labeled as a *risky* packet by the packet labeler, the packet is immediately sent to a buffering system which feeds it into a simple proxying mechanism. This proxy module is a transparent proxy, ensuring the switch queries the controller for only those senders that complete TCP handshake. This proxying of the connection helps in filtering out TCP handshake packets with spoofed source address since the attacker cant complete a TCP handshake with a spoofed address.

VI. Evaluation

A. *Regular Traffic*

The network was tested by visiting benign links and taking traffic measurements with and without our solution. For the purpose of experiment a local web server was created, running a simple Blog Website. Website was created having multiple links and multiple host were made to access this website. Our module was able to detect and prevent DDoS attacks with an accuracy of 97.31% with a computational overhead of 10.63%

B. SYN Flood Attack

For the simple ICMP ping flood attack, blocking all ICMP packets at the switch immediately stopped the attack. The second approach that was mentioned, selectively responding to ping requests, also helped in mitigating an ongoing attack. The deciding factor, or the selectiveness ranged between 0 (dropping all the packets; equivalent to all ICMP packets blocking) and 1 (allowing all the packets; equivalent to no protection). The switch can make decision depending upon the traffic needs dynamically. The observations are shown in Table 1.

TABLE I
PERFORMANCE IN SYN FLOOD ATTACK

Implementation	***Success Rate***	***Overhead***
Simple OpenFlow	0%	-
OpenFlow with our Module	98.78%	10.81%

C. Resource Exhaustion Attack

A decision making system was designed that labeled the DDoS packets into individual classes, safe, risky and dangerous. The system runs independently on a switch and labels the packet. To finally test the system different subsets of the KDD labeled database [16] was taken. The system was able to highlight the DoS packets with an accuracy of 91.1% . The false positives were restricted to 6.2%. The True Negatives and False Negative were found to be 93.5% and 8.3% respectively.

TABLE II
PERFORMANCE IN RESOURCE EXHAUSTION ATTACK

Implementation	***Success Rate***	***Overhead***
Simple OpenFlow	0%	-
OpenFlow with our Module	91.1%	6.38%

VII. Conclusion and Future Work

This work introduces a simple but dynamic decision making system that can be deployed at SDN switches to filter out the packets taking part in a DoS attacks. The system takes into account multiple packet characteristics to decide whether a packet is malicious or not. If the score of a packet crosses the danger threshold, that packet is dropped outwardly, if the packet's score is below the safety threshold then it is allowed to pass. Packet falling in the risky category were buffered and proxied unless they completed their handshake. A future extension for this work would be to perform this real time packet monitoring and filtering at the SDN controller level. The controller will push these packet filtering decision in real time to the switches for filtering attack packets. Another enhancement that can be done would be to perform deep learning to identify more advanced DoS attacks that may have extensively disguised.

References

[1] Sheng, Zhengguo, et al. "A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities." IEEE Wireless Communications 20.6 (2013): 91-98.
[2] Sezer, Sakir, et al. "Are we ready for SDN? Implementation challenges for software-defined networks." IEEE Communications Magazine 51.7 (2013): 36-43.
[3] Feamster, Nick, Jennifer Rexford, and Ellen Zegura. "The road to SDN: an intellectual history of programmable networks." ACM SIGCOMM Computer Communication Review 44.2 (2014): 87-98.
[4] Jarraya, Yosr, Taous Madi, and Mourad Debbabi. "A survey and a layered taxonomy of software-defined networking." IEEE communications surveys & tutorials 16.4 (2014): 1955-1980.
[5] Chen, Tao, et al. "Software defined mobile networks: concept, survey, and research directions." IEEE Communications Magazine 53.11 (2015): 126-133.
[6] Hong, Sungmin, et al. "Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures." NDSS. 2015.
[7] Kapetanovic, Dzevdan, Gan Zheng, and Fredrik Rusek. "Physical layer security for massive MIMO: An overview on passive eavesdropping and active attacks." IEEE Communications Magazine 53.6 (2015): 21-27.
[8] Shin, Seungwon, et al. "Avant-guard: Scalable and vigilant switch flow management in software-defined networks." Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013.
[9] Ambrosin, Moreno, et al. "Lineswitch: Efficiently managing switch flow in software-defined networking while effectively tackling dos attacks." Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security. ACM, 2015.
[10] Mirkovic, Jelena, et al. "Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Computer Networking and Security)." (2004).
[11] Long, Neil, and Rob Thomas. "Trends in denial of service attack technology." CERT Coordination Center (2001).
[12] De Oliveira, Rogrio Leo Santos, et al. "Using mininet for emulation and prototyping software-defined networks." Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on. IEEE, 2014.
[13] Eddy, Wesley M. "TCP SYN flooding attacks and common mitigations." (2007).
[14] Mousavi, Seyed Mohammad, and Marc St-Hilaire. "Early detection of DDoS attacks against SDN controllers." Computing, Networking and Communications (ICNC), 2015 International Conference on. IEEE, 2015.
[15] Perez, K. Guerra, et al. "A configurable packet classification architecture for software-defined networking." System-on-Chip Conference (SOCC), 2014 27th IEEE International. IEEE, 2014.
[16] Bay, Stephen D., et al. "The UCI KDD archive of large data sets for data mining research and experimentation." ACM SIGKDD Explorations Newsletter 2.2 (2000): 81-85.