Leveraging Inter-Controller Communication to MitigateDDoS Attacks in SDN Networks

J.P. Houle
Dept of CEE & NPSIA
Carleton University

S. Ahmadi
Dept of SCE
Carleton University

B.C.A. Robart
School of EECS
University of Ottawa

A. Matrawy
Carleton School of IT
Carleton University

Abstract - **In this poster, we propose leveraging inter-controller communication between two or more controllers in Software Defined Networks (SDNs) to inform other controllers about potential attacks so that they can proactively apply a mitigation strategy. We demonstrate and measure the effectiveness of our method by running a series of tests in an emulated network. We analyze our test results in terms of reducing the overall detection interval for a SDN with multiple controllers. Our testing indicates that inter-controller communication allows pre-emptive mitigation of Distributed Denial of Service (DDoS) vectors.**

Index Terms - Software Defined Network; SDN; inter-controller communication; DDoS; OpenFlow; RYU

I. Introduction

The specific threat model examined in this poster involves a large SDN (Fig 1) that consists of multiple SDN domains where each domain is managed by a controller. The attack scenario involves malicious traffic entering through a switch in domain A to flood a certain host within the same domain. The switch is controlled by SDN controller C1, which has been enabled to detect DDoS attacks. After a successful detection (taking one detection interval), the attack traffic going to the target host in domain A will be dealt with as long as it continues to ingress through a switch controlled by C1. Subsequently, the attack traffic destined for the target host within domain A takes a different network path and enters the network through a switch within domain Z controlled by SDN controller C2. The offending traffic will trigger the DDoS detection process of C2 and, after another detection interval, will be detected and mitigated. At this point, the target host within domain A will have experienced attack traffic over two detection intervals, the first when the traffic entered through the domain A switch and the second when the traffic entered through the domain Z switch.

A review of the relevant literature [1, 2, 3, 4, 5] did not produce an example of inter-controller communication to address attacks spanning multiple SDN controllers. This poster proposes a solution that leverages simple communications between controllers to help reduce the impact of the attack by eliminating one of the detection intervals in this scenario. When the controller in domain A detects the malicious traffic, it notifies all other controllers in the network so that they can proactively implement mitigation measures and eliminate their detection intervals.

II. Proposed Solution

In our proposed method, when an attack is detected by a SDN controller, it sends the information about the attack to other controllers in the network. We call this "*inter-controller communication*". When attack information is shared between controllers, the detection interval for all of them other than the first is eliminated and the traffic is mitigated immediately.

Our implementation has three main components: a DDoS detection method; an attack mitigation mechanism; and, an inter-controller communication (ICC) technique. For DDoS

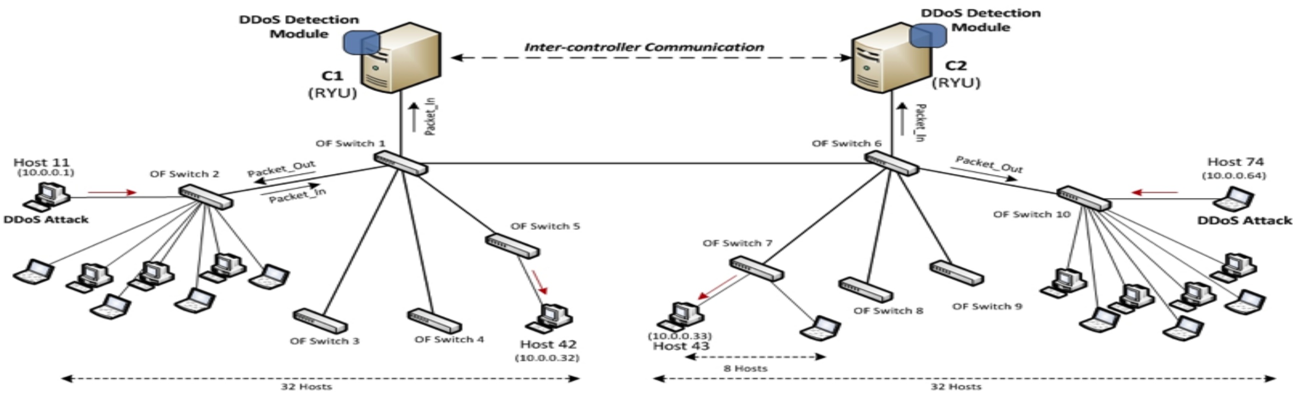

Figure 1 - Multi-domain SDN with DDoS detection and inter-controller communications enabled in SDN controllers

detection, we used work by Mousavi and St.-Hilaire that demonstrated an entropy-based detection algorithm [6, 7]. The entropy method detects abnormalities in the distribution of destination IP addresses. The original work focused on attack detection but did not mitigate the attack. We made some adjustments to the method so that we could identify individual IP addresses under attack and implement our mitigation mechanism.

Our mitigation mechanism balances the need to reduce the impact of malicious traffic while still allowing legitimate traffic to reach its destination. To this end, we configured quality of service (QoS) queues that rate-limit all traffic of the same type (TCP or UDP) as the attack traffic and is destined to the IP address under attack.

The ICC method used TCP sockets to share information between controllers. The controllers used this information to proactively insert flow table entries that mitigate the malicious traffic. This eliminates the detection interval of the other controllers for a known attack. Note that we assume that controllers are trusted. The way controllers implement trust mechanisms is outside the scope of this work.

III. Test Results and Analysis

We used Mininet [8] to emulate our SDN environment and a Kali Linux VM with Scapy [9] to produce attack traffic. Our environment is visualized in Fig 1. We used RYU [10] for the SDN controllers as it supported rate-limited queues [11] and all switches are OpenFlow SDN switches. There are 64 emulated hosts, 32 in each domain. We modified the RYU "simple switch" sample code to implement the mitigation strategy proposed in section II.

Test	Attack Rate	Mitigation	ICC
Test 1a	Low	Enabled	Disabled
Test 1b	Medium	Enabled	Disabled
Test 1c	High	Enabled	Disabled
Test 2a	Low	Enabled	Enabled
Test 2b	Medium	Enabled	Enabled
Test 2c	High	Enabled	Enabled

Table 1 - Test Characteristics Summary

We present two of the tests that were executed to assess our proposed solution. Tests 1 and 2 (Table 1) were run at three different attack rates: (a) low; (b) medium; and, (c) high. Test 1 demonstrated the attack detection interval in packets required to detect an attack and Test 2 showed the impact of introducing ICC. Both tests used the same attack scenario where the attack traffic point of origin changes but the target host remains the same.

A summary of the preliminary results for these tests are shown in Fig 2. The figure compares attack detection intervals in terms of number of packets needed to detect for both controllers.

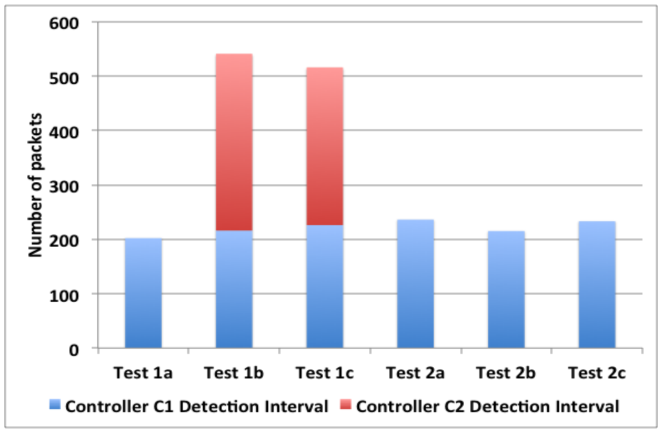

Figure 2 - SDN Controller Detection Interval Results

Test 1a shows a result where C1 detects an attack but C2 does not. This could be due to the low rate of attack traffic. This means that when the attack traffic origin shifts to C2's domain, no traffic mitigation is performed and the attack continues unimpeded. Subsequent tests at higher attack rates (1b and 1c) show that C2 detects an attack and mitigates it based on its own detection interval. An important point to note is that controller C2 is exposed to attack traffic that has already been identified by another controller.

Tests 2a-c introduce ICC that is used to share information about detected attacks between the two controllers. The results show that the detection interval on C2 is eliminated due to ICC. The attack traffic is immediately mitigated because C2 recognizes it based on the information shared by C1. Note that test 2a shows mitigation of attack traffic that was not detected by C2 in test 1a. This method enables the controllers to protect themselves by pooling their capabilities to detect attacks.

References

[1] "A Survey of Securing Network Using Software Defined Networking", Ali Et Al., IEEE Transactions on Reliability, Vol. 64, No. 3, Sept 2015
[2] "Detecting Distributed Denial of Service Attacks: Methods, Tools and Future Directions", Bhuyan Et Al., Oxford University Press 2013.
[3] "Role-Based Multiple Controllers for Load Balancing and Security In SDN", Chourishi Et Al., Proc. of IEEE (IHTC), 2015.
[4] "A Feasible Method to Combat Against DDoS Attack in SDN Network", Dao et Al., IEEE International Conference on Information Networking (ICOIN), 2015.
[5] "A SDN-Oriented DDoS Blocking Scheme for Botnet-Based Attacks", Lim et al., IEEE Sixth International Conf on Ubiquitous and Future Networks (ICUFN), 2014.
[6] "Early Detection of DDoS Attacks in Software Defined Networks Controller", Mousavie S., Master's Thesis, Carleton University, 2014.
[7] "Early Detection of DDoS Attacks Against SDN Controllers", Mousavie and St-Hilaire, Proc. of ICNC 2015.
[8] (2015, Oct) Mininet. [Online]. http://Mininet.Org/
[9] (2015, Oct) Scapy. [Online]. http://www.Secdev.Org/Projects/Scapy/
[10] (2015, Oct) RYU Documentation. [Online]. https://Osrg.Github.Io/Ryu-Book/En/Html/
[11] (2015, Oct) Openflow V1.3 Documentation. [Online]. https://www.Opennetworking.Org/Images/Stories/Downloads/Sdn-Resources/Onf-Specifications/Openflow/Openflow-Spec-V1.3.0.Pdf