Selective Packet Inspection to Detect DoS Flooding Using Software Defined Networking (SDN)

Tommy Chin Jr
Rochester Institute of Technology
Rochester, New York, USA
txc9627@rit.edu

Xenia Mountrouidou
Jacksonville University
Jacksonville, Florida, USA
xmountr@ju.edu

Xiangyang Li
Johns Hopkins University
Baltimore, Maryland, USA
xyli@jhu.edu

Kaiqi Xiong
Rochester Institute of Technology
Rochester, New York, USA
kxxics@rit.edu

***Abstract*—Software-defined networking (SDN) and OpenFlow have been driving new security applications and services. However, even if some of these studies provide interesting visions of what can be achieved, they stop short of presenting realistic application scenarios and experimental results. In this paper, we discuss a novel attack detection approach that coordinates monitors distributed over a network and controllers centralized on an SDN Open Virtual Switch (OVS), selectively inspecting network packets on demand. With different scale of network views and information availability, these two elements collaboratively detect signature constituents of an attack. Therefore, this approach is able to quickly issue an alert against potential threats followed by careful verification for high accuracy, while balancing the workload on the OVS. We have applied this method for detection and mitigation of TCP SYN flood attacks on Global Environment for Network Innovations (GENI). This realistic experimentation has provided us with insightful findings helpful toward a systematic methodology of SDN-supported attack detection and containment.**

I. Introduction

Recently software-defined networking (SDN) [1] has attracted a lot of attention for its potential to provide new operational models of security services and applications. Its capability to separate control from data flows is a natural aid for the flexibility to configure security devices and invoke new services on demand. Many studies of information security topics on OpenFlow API protocols and SDN can be categorized into two groups, i.e. security for SDN that seeks to strengthen its functionalities, and SDN-supported security services and applications with one focus on new attack detection and mitigation schemes.

However, studies on SDN-supported attack detection and mitigation so far lack real experimentation and implementations. Many such studies simply propose a vision of what SDN should be capable to support and how the architecture supported by SDN may look like in a security service, such as in [3]. A few studies [4], [5], which actually experimented their proposed ideas, are very limited in realizing the potential of SDN. The best arguments that these studies have made are that more data are made available if needed for attack detection and traffic reshaping is feasible due to the OpenFlow protocol [2]. Additionally, one significant challenge facing these studies is their reliance on either SDN hardware, not easy to configure and scale up, or software simulators, limited in faithfulness of supporting realistic networking scenarios.

Two studies are especially relevant to the challenges and solutions that we consider in this study. The OrchSec architecture uses decoupled monitors and SDN correlators in order to mitigate different types of attacks [3]. The authors have examined different decoupling designs with the help of a so-called orchestrator facility. They claim that these strategies can have better resolution and analysis capabilities in detecting attacks for improved performance. However, the minimal experimentation over POX and Floodlight correlators with Mininet simply looks for traffic irregularities. The NICE framework is an IDS agent to monitor mirrored traffic and to propose potential countermeasures to attacks [4]. The detection is based on attack graphs of known system vulnerabilities by a so-called attack analyzer. It is fairly close to the traditional distributed IDS where a central correlation manager works with distributed sensors that collect network traffic with support by OpenFlow. Again, this paper is mainly on the overall framework, lacking many specifics in protocols and applications.

In this study, we explore a collaborative approach that can employ a system of sensory monitors distributed over a network and centralized correlation functions at Open Virtual Switch (OVS) nodes. A monitor is sensitive and lightweight to quickly detect anomalies in network traffic. A close-by correlator, upon receiving an alert from a monitor, verifies the suspicion by inspecting detailed evidence of network packets in its neighborhood for additional attack signatures. Consequently, the correlator, the monitor, as well as the OVS controller coordinate further actions to either update normal traffic profiles or mitigate impacts of an attack. In this way, the workload on the controller is balanced by being selective and informed in inspecting network traffic only on demand, which is critical to large-scale networks. Moreover, this study demonstrates the capability of this scheme with the help of Global Environment for Network Innovations (GENI), a virtual testbed that fully supports SDN for large, at scale experimentations. We test this new approach against the TCP SYN flood attack for its effectiveness.

II. GENI and SDN

Global Environment for Network Innovations (GENI) has become an emerging virtual infrastructure which leverages networked computing technologies for at-scale experimentations [**?**]. GENI is an NSF-sponsored infrastructure. Heterogeneous GENI resources permit users deep programmability throughout the network and are shared among multiple users. Compute, storage and networking resources provisioned in concert (as slices) to support repeatable experimental activities. GENI

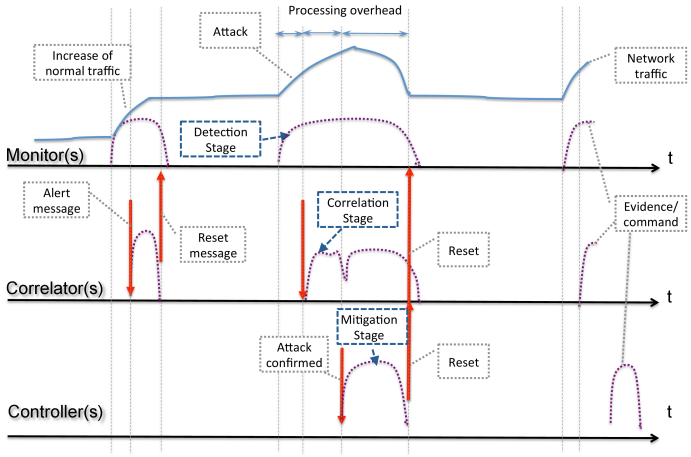

Fig. 1: A pulse-like work cycle where monitors distributed over the network raise an alert that triggers correlators for active evidence collection/correlation and, if needed, mitigation.

is an ideal infrastructure for many research and educational experimentations. ExoGENI and InstaGENI are fully equipped with SDN [**?**].

III. A COLLABORATIVE ATTACK DETECTION AND CONTAINMENT APPROACH

DoS flooding attacks have been a growing problem for computer networks and Internet users [7]. These attacks try to deplete the recourses that a victim device has, being the network bandwidth or host resources. They utilize a variety of techniques of flooding, amplification, protocol exploiting, and malformed packets. Various defense mechanisms have been proposed against these attacks [8]. Recently SDN has been applied in several studies as reviewed in previous sections. However, DoS still poses many significant challenges to current detection and correlation solutions:

The ability of Intrusion Detection Systems (IDS) is limited by what they see in data and thus by their locations on a network. Ideally, an IDS should have a capability to log and analyze every piece of data on the network. Obviously this is not realistic and a good solution must seek to allow distributed elements of IDS to work together.

Even if a network element, such as a SDN correlator, is able to get hold of every bit of data, it is impossible to do deep inspection on every network packet. Information collection and processing overheads have to be serious concerns for any bottleneck device or centralized facility. We must make very careful use of such resources.

There is always a tradeoff in handling each of these challenges. For example, an IDS sensor placed close to a botnet employed by DoS attacker may be effective in identifying the source/path of attack; another IDS in the neighborhood of a targeted victim may be able to quickly detect anomaly in the traffic, e.g. increase in volume. Therefore the key is to consider how to best utilize different IDS elements together in a dynamic and adaptive way. SDN is a natural fit to promote such operational capabilities. A critical observation of DoS

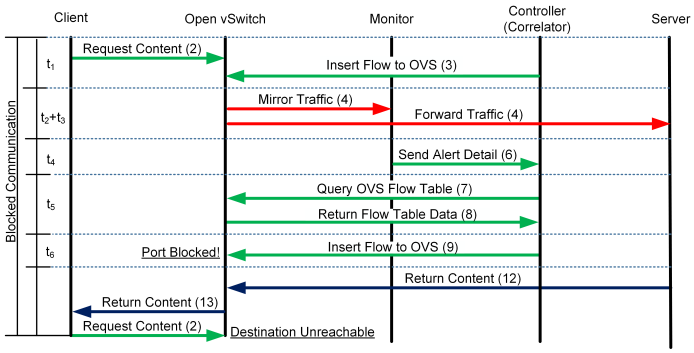

Fig. 2: The communication protocol between the Monitor, the controller (Correlator), and other elements.

attacks (and many others) is that usually multiple constituents of different characteristics together define the unique signature of one special type of attack. Consider the TCP SYN flood attack studied here. The first pattern is a surge of traffic volume of SYN requests because this type of attack tries to deplete the network resource of connection buffer at the victim computer. Setting up a control threshold over the baseline traffic can quickly recognize such a surge. Unfortunately normal network activities can shift. This first pattern by itself does not accurately detect attacks and relying on just that, would unnecessarily generate many false alarms. The second symptom of SYN flood attack, i.e. spoofed source IP addresses in invalid SYN requests necessary to prevent connections from being closed too soon, becomes very informative. However to recognize these fake IP addresses requires adequate knowledge of the network topology and configuration to be able to tell them apart from other legitimate IPs. Moreover, to precisely correlate attack traffics to reveal rogue nodes and attack paths, data in those packets of fake IP addresses have to be assembled and dissected as needed. Fortunately such knowledge is available to SDN controller. Once this symptom is seen we can confidently say SYN flood is in presence.

As shown in Figure 1, the work cycle of this collaborative approach involves one or multiple monitors that constantly observe the network traffic, and correlators that only respond to the alerts from monitors on demand. Once an alert is received, i.e. surge of TCP SYN packets that exceeds the control threshold established on the normal traffic baseline, a correlator immediately takes actions based on the type of alert. Very often, it is necessary for the correlator to first actively collect additional packets for pertinent information, according to a sampling scheme over the nodes on a SDN network. If after close examination of these evidences and the flow-forwarding table, the correlator cannot identify any invalid IP addresses, this may indicate simply a flash of normal transactions. Then a reset command is issued to the reporting monitor(s). Normal traffic profile can be updated as necessary to establishing a current baseline. This effectiveness achieved by the separation of quickly raising alerts and close investigation, is largely at the cost of communication overhead between monitors and correlator.

IV. SYSTEM ARCHITECTURE

In this section we describe the implementation of the collaborative detection and containment scheme.

A. System Design

In Figure 2, a Client is a computer node with either malicious intent or a regular user generating traffic to utilize resources within a network. A Server is a device running a web server. It is assumed that multiple Client nodes may exist in the system. Without loss of generality we assume that there is only one web Server under attack. The two most important features of the depicted logical architecture are the Monitor and the Correlator. The latter is hosted by an SDN controller. There may be multiple Monitor and Correlator nodes, each corresponding to a specific network. The Monitor has a dual functionality. First, it operates as a sensing node, using an IDS. Second it acts as a messenger sending alerts and additional information on a potential attack to a close by correlator. The Correlator correlates the alert from the monitor to the information maintained by the OVS switch for deep packet inspection. It verifies if there is an attack in the network, identifies the port and computer from which the attack traffic is coming, and mitigates the attack by dropping the flow.

B. Algorithms

The Monitor is running a real time communicator algorithm, i.e., a lightweight implementation that manages alerts and additional information of relevance. It listens continuously for alerts. Once an alert is issued by the IDS, the Monitor notifies the associated Correlator by a simple attack flag that corresponds to a specific type of attack. Then the monitor collects a minimum amount of information related to the attack and sends it to the correlator (label 6). Specifically, for SYN flood attacks, it sends out a number of source IP addresses found in SYN packets.

The Correlator, hosted on the SDN controller, runs another algorithm that maintains an open communication with the monitor and the OVS, and fulfills three tasks related to the correlator and the controller. First, the presence of an attack is to be verified after an alert of a specific attack type is received. For SYN flood attacks, a hash table based on the original flow table of the OVS switch is created with the unique port numbers as keys and the IP addresses as data. The Correlator queries this table using the IP addresses from the monitor to look for any unknown IPs, in order to confirm the attack. Second, once the attack is confirmed, the Correlator performs additional queries to a second hash table created based on the current flow table. Finally, using the OpenFlow API the correlator issues a rule directing to drop the rogue traffic.

C. Communication Protocol

Communication between the architectural components is crucial to our synergistic approach on SDN. The communication protocol is shown in Figure 2. Initially the OVS will communicate with the Controller to allow the adaptation of network flow. When a User or Attacker (Client) issues a request to the targeted Server, a flow rule is issued (3) to the OVS from the Controller. As the request (2) is sent to the destination Server, it is also mirrored (4) to the monitor for inspection. Once the Server receives the request, the content is then returned to the Client through the OVS. Up to this point if no alert is raised, the traffic can be considered normal. If an alert is raised and sent (6) to the Correlator, upon receipt it will query the dump-flows function of the OVS (7) in order to collect information relating to port number, mac-addresses, and various IP addresses that belong to the topology. The OVS will reply with the flow tables (8) needed to confirm there is an attack and then correlate the attack information with a port. Finally, the Controller will send a command to the OVS (9) to take mitigation actions if there is attack traffic.

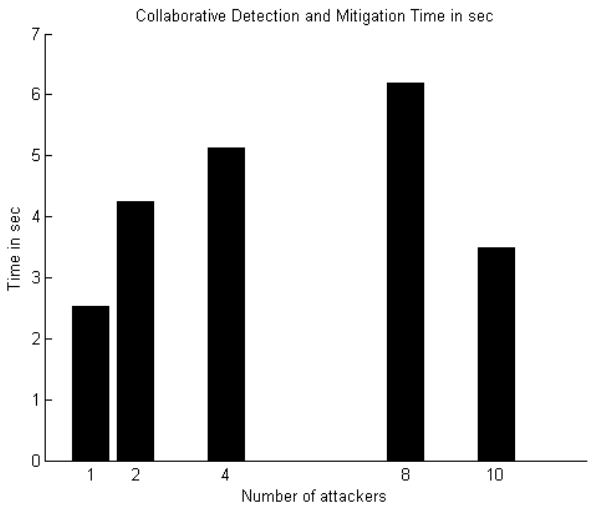

Fig. 3: Total time of detection and mitigation.

V. EXPERIMENTATION

A. Implementation on GENI

The implementation on the GENI cloud infrastructure offers the opportunity to validate the assumptions of our design as well as to demonstrate the efficiency and accuracy of our collaborative scheme. All the host nodes, controllers and monitors are implemented using XEN VMs that run Ubuntu Linux 64 bit, with Intel(R) Xeon(R) processors at 2.10GHz. The traffic generators create normal traffic with iperf, a commonly used network testing tool that measures performance, to the full capacity of the network bandwidth. A tool called hping3 is used to implement TCP SYN floods with IP address spoofing. Attack traffic is commonly characterized by a rate of a few thousands of SYN packets per second.

The correlator is running the correlation algorithm together with a Pox controller that controls the OVS. Snort is used to implement the IDS at the monitor. The mechanism that issues alerts about a potential attack is implemented by a Snort rule. A control threshold on the intensity of TCP SYN packets is selected through empirical analysis of normal traffic. If this threshold is exceeded, the IDS raises an alert and communication is initiated between the monitor and the correlator. The communication protocol between the monitor and correlator as well as the correlation algorithm are implemented in Python using socket programming.

B. Result Analysis

The goal of our experimentation is to analyze the performance of the collaborative detection and mitigation scheme using standard metrics. The performance is measured by the total duration of detection and mitigation time, starting from

the moment that the attack is launched until the time the mitigation scheme is fully in effect. As shown in Figure 2, this duration is calculated using a sum of time intervals: , where t_1: the time it takes for the mirrored traffic to reach the monitor; t2: the constant time, i.e., 1 second, it takes for the monitor to process packets and raise an alert based on a specific control threshold; t3: the time it takes for the monitor.s alert to reach the correlator; t4: the time needed for the correlator to process the monitor.s alert; t5: the time it takes for the correlator to query the OVS switch; and t6: the time needed for the controller to transmit proper mitigation commands to the OVS. These time intervals depend either on the network speed, e.g., for t_1, t3, and t5, or a machine.s processing power, e.g., for t4, or both, e.g., for t6. We analyze the performance of the collaborative detection by itemizing it to individual intervals in order to focus on bottlenecks for useful insights.

We have experimented with different scenarios of regular traffic, where the traffic load was ramped up while attack traffic remained constant. This was achieved by simulating multiple parallel clients running using iperf. We have scaled up from 10 to 60 simultaneous clients, which is the maximum that our current infrastructure can support. Figure 3 shows that the collaborative detection and mitigation mechanism scales well, as the user traffic ramps up and is multiplexed with flood traffic that remains constant. The total time it takes from the time of the attack until the mitigation action occurs as shown in Figure 6, is less than proportional to the increase in the regular traffic in the network.

Figures 4 and 5 show the individual intervals, t_1, , t6, that affect the total time for synergistic detection and Figure 3: Total time of detection and mitigation. These figures demonstrate how the performance of different parts of our implementation, i.e., monitor, correlator, and OVS, scales when the regular traffic is ramped up from 10 up to 60 parallel clients. We have divided these intervals to the parts that are affected by the monitor processing time and its communication with the correlator (Figure 4) and the parts that are affected by the correlator processing time and its communication with the OVS (Figure 5). It can be observed that the monitor is the bottleneck in our experimentation, as it takes considerable time to process the packets and raise an alert for suspicious traffic. Interestingly, the most time consuming part of the monitor implementation the time it takes for the monitor to receive the attack packets, especially as the traffic in the network is ramped up. The processing time to raise an alert is constant, equal to 1 second. To improve the performance of our current implementation we will need to either increase the number of monitors or use more powerful machines. Surprisingly, for the correlator the most time consuming part of the implementation is not the query to the OVS as expected (t4), but the processing of the OVS data (t6). To resolve this issue we plan to investigate more efficient implementation algorithms for detecting the attacker.

VI. Conclusion and Future Work

We have presented the implementation of a collaborative detection and containment mechanism of network attacks. Our approach is unique to use a synergistic monitoring, detection, and mitigation strategy to realize the full capabilities of SDN. The experimentation on GENI has shown that our solution

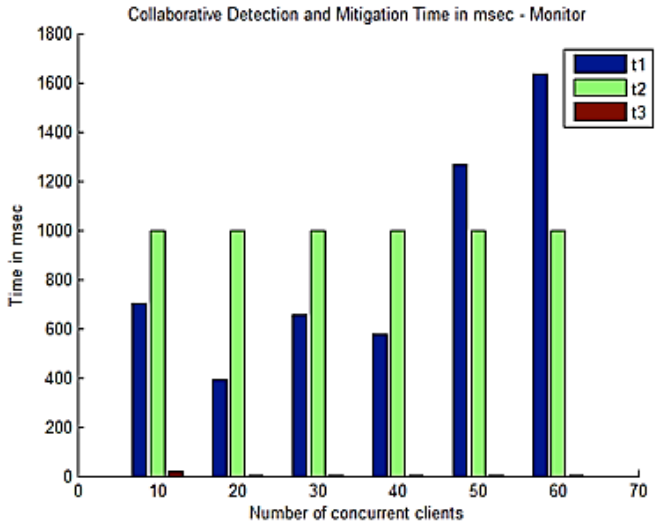

Fig. 4: Monitor Performance

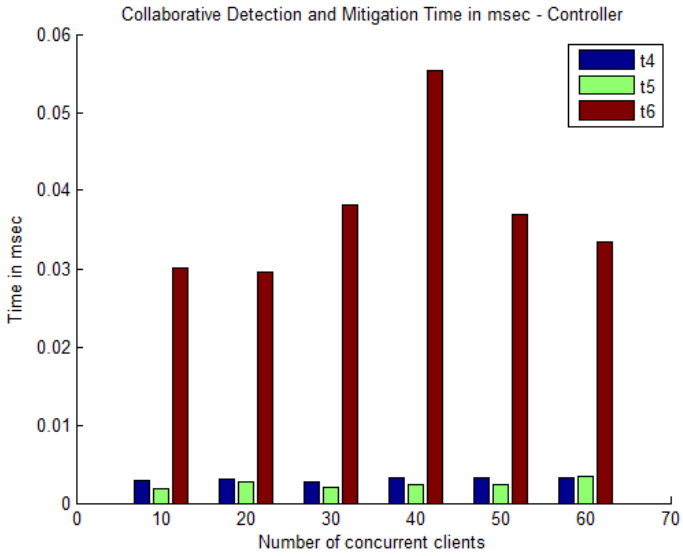

Fig. 5: Correlation Performance

is scalable to process high volume of traffic and large scale attacks.

We will continue to test our approach for more results of its effectiveness and efficiency. In addition, we are working to apply this collaborative approach to other security applications, including detection and mitigation of covert channels. Our goal is to develop a systematic methodology along this line of work.