Distributed SIP DDoS Defense with P4

Aldo Febro
School of Computer Science
University of Hertfordshire
Hatfield, UK AL10 9AB
Email: a.febro@herts.ac.uk

Hannan Xiao
School of Computer Science
University of Hertfordshire
Hatfield, UK AL10 9AB
Email: h.xiao@herts.ac.uk

Joseph Spring
School of Computer Science
University of Hertfordshire
Hatfield, UK AL10 9AB
Email: j.spring@herts.ac.uk

***Abstract*—SIP DDoS attack is growing and has a real threat to crippling public communication infrastructure. The standard approach to building the defense is at or near the attack destination (i.e. victim's location). This approach is struggling to keep up with the growing volume and attack sophistication. To be better prepared for future attacks, the workload needs to be distributed, and the attack needs to be mitigated as close to the attack source as possible. This paper experiments with data plane programming (P4) and control plane programming of Ethernet switches to provide first-hop detection and mitigation capability for SIP INVITE DDoS attack at every switchport. This approach creates a distributed or source-based defense component which could be added to the existing destination-based components to create a more comprehensive overall solution that is extensible, economical, and scalable against SIP DDoS attack of the future.**

***Keywords*—SIP, DoS, DDoS, SDN, P4, data plane, control plane**

I. Introduction

Application layer DDoS attack against SIP has the potential to become a major public security issue when it is targeted towards critical public infrastructure like Next Generation 911 (NG9-1-1) system [1]. A recent empirical study shows that 200 REGISTER messages/second and 110 INVITE messages/second was enough to cause Denial-of-Service [2].

Current approach towards SIP DDoS defense deployment leans towards the destination-based approach [3] where defense resources are deployed at the destination of the attack i.e. at the victim's location. With this approach, the potential victims need to keep building bigger and more sophisticated defense system to keep up with the growing DDoS attack. Considering that such a system is not trivial to operate and upkeep, a new design needs to be explored to enhance existing approach.

In contrast with the destination-based approach, the source-based approach is deployed near the source of the attack [3]. With this method, SIP DoS packets are detected and mitigated at the point-of-entry to the network. As these attacks were dealt when the volume was still negligible, it did not require much CPU and memory resource on the network equipment (router or switch) to process these packets. This approach requires SIP DDoS defense capability to be present at the network edge to form a distributed approach towards SIP DDoS defense.

This paper describes a novel approach that uses P4 [4] data plane programming and control plane programming to create distributed approach for SIP DDoS defense. A P4-compatible switch provides a rich filtering expression that allows each switchport to perform deep packet inspection for SIP DoS detection. For example, the switch can be programmed to count the number of incoming SIP REGISTER or INVITE packet sent by the directly connected device. When the threshold is exceeded, P4-controller instructs P4-switch to drop the packet and therefore mitigate the attack right at the source.

A. Related Work

A comprehensive survey paper on VoIP security research indicates that half of DoS-related vulnerabilities are on the endpoints [10]. Hussain et al. (2015) in their comprehensive study of flooding attack wrote that SIP INVITE message is considered as one of the major root causes [7], and application-level DDoS flooding attacks like SIP remain to be an open problems [12]. Some of the related works that use destination-based as well as source-based approach are as follows:

Ormazabal et al. (2008) [5] introduced carrier class, CAM-accelerated solution for SIP DoS defense called "Secure SIP." It proposes the use of 2 filters (SIP-filter and dynamic pin-hole RTP-filter). It handles spoofed packets by using Return Routability filter. SIP method-based filters and state validation filters are used to provide protection against SIP protocol vulnerability against DoS attack.

Huici, Niccolini, and DHeureuse (2009) introduced "SIP Defender" where they proposed source-based attack mitigation coupled with destination-based attack detection [6]. There are two Special filtering box called SIP Controller (SC). SC1 is installed at source-network, while SC2 is installed at destination-network. At the destination network, IDS is also assumed to be in place. When the IDS detects an attack, it will send a list of SIP users to SC2. SC2 then send a filter request to SC1 which will install and mitigate the attack.

Liu, Yin, and Lee (2014) [8] proposed source-based attack mitigation approach for traffic coming from untrusted peer network using Cloud SIP FireWall (Cloud SFW) and SDN controller. The proposal uses queuing theory to construct appropriate low-priority and high-priority queue at Cloud SFW. Cloud SFW can monitor bandwidth usage and notify SDN controller when it hits a certain level. During an attack, SDN controller will coordinate mitigation effort with either local Cloud SFW or remote Cloud SFW installed at the untrusted peer network.

	Source-based detection	Source-based mitigation	Destination-based detection	Destination-based mitigation
Secure SIP[5]	N	N	Y	Y
SIP Defender [6]	N	Y	Y	N
Cloud SFW[8]	N	Y	Y	Y
This paper	Y	Y	N	N

TABLE I: Destination-based vs. source-based defense

B. Research Gap

Lack of first-hop detection and mitigation. There are several advantages to detect and mitigate at the first network hop. First, the amount of traffic is still small and therefore can be mitigated by the first router or switch. Second, bandwidth is saved by not transporting attack traffic all the way to the destination only to be dropped or to be used for malicious purposes. Third, limit collateral damage along the way towards the victim.

Limited number of sensors. Sensor detects and recognizes SIP DoS attack pattern within SIP transaction or dialog. To recognize application layer attack like SIP, the sensor need to know how the protocol works and how a session deviates from normal pattern. Most proposals have too few sensors which restricts the visibility of ongoing attack and therefore not having early warning. The cost and complexity of sensor deployment are usually the factors that make it not practical to have more sensors.

Limited number of mitigation points. There are only selected nodes in the network that can perform packet filtering. The attack could have grown to a point where it requires more resources than what is currently available to deal with the attack.

Researchers did not generally consider source-based DDoS defense approach due to the cost involved and complex deployment issues. Networking vendor produces Ethernet switches in mass quantities with just enough hardware resources to support selected features. As a result, the features of these switches are fixed and not extensible by end-users. For specific functions, end-users would have to get special-purpose network appliance e.g. load balancers, firewall, VPN, etc. Procuring these appliances for mass deployment is not practical and financially justifiable.

However, several major trends that change the market: the demand for faster innovation, the rise of Software Defined Networking (SDN), and the trend of open source software and hardware movement. Competitive global business market demands for faster innovation and production cycle. Organizations view traditional networks as not suitable for the needs of today. SDN promises to enable innovation through network programmability [9] while Open source networking hardware provides the platform to run these software and to provide physical connectivity. These trends would potentially make it affordable for end-users to replace their existing fixed-function conventional switch with a programmable switch that matches or exceeds the performance of the conventional switch.

C. Research Contribution

To the best of our knowledge, this study made the following novel contributions: SIP DDoS detection through the use of data plane programming of Ethernet switch, First-hop SIP DDoS attack detection and mitigation capability for every port on the Ethernet switch, and Distributed SIP DDoS defense approach using Ethernet switches at the network edge.

II. Proposed Model and Methods

The proposed model was built to simulate typical network where SIP clients are connected to access-layer switches as depicted in Figure. 1. Each of these P4-switches are controlled by a P4-controller over TCP socket connection. Each port on these switches was programmed to perform deep packet inspection for INVITE or REGISTER packets. There was a counter associated with each port to count the number of INVITE or REGISTER packets received. Every second, P4-controller evaluated this counter to determine whether SIP INVITE flood attack took place. When P4-controller had determined that an attack took place, it sent a command to P4-switch to drop subsequent INVITE or REGISTER packets from that particular port. When the attack had stopped for one consecutive minute, P4-controller sent a command to P4-switch to resume to normal processing mode.

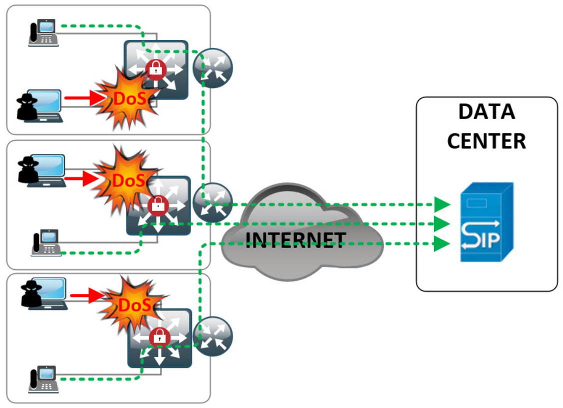


Fig. 1: Distributed & source-based SIP DDoS defense at the network edge

A. Test environment

To experiment with the above model, an *Ubuntu* virtual machine with 2GB RAM and 2 CPU is used to run *mininet* [18] to create three virtual hosts and three virtual switches topology. The SIP proxy (10.0.0.1) is attached to switch-1 at port-1, the attacker (10.0.0.2) is attached to switch-2 at port-2, a legitimate user (10.0.0.3) is attached to switch-3 port-3. The controller is running on the host machine and communicate with three switches via socket connections (port TCP/22221 for switch-1, TCP/22222 for switch-2, and TCP/22223 for switch-3). This environment is depicted in Figure. 2.

A P4-compatible virtual switch called *bmv2*[15] is used instead of default virtual switch that comes with *mininet*. Useful topology examples are available [16]. A P4-configuration generator called *p4c-bm*[17] is used to convert P4 code to JSON configuration file that is required by *bmv2*.

For P4-controller, a python script was used to send commands and interact with P4-switches e.g. check SIP counter, drop packet, resume to normal operation.

SIPp is a SIP packet generation tool that was used to generate SIP traffic. For SIP proxy, it is set to run in User Agent Server (UAS) mode, whereas for SIP clients, it is in User Agent Client (UAC) mode.

Packet capture (pcap) utility tool *tshark* is used to verify packets sent and received by the SIP Proxy (at switch-1:port-1) and by the SIP clients (at switch-2:port2).

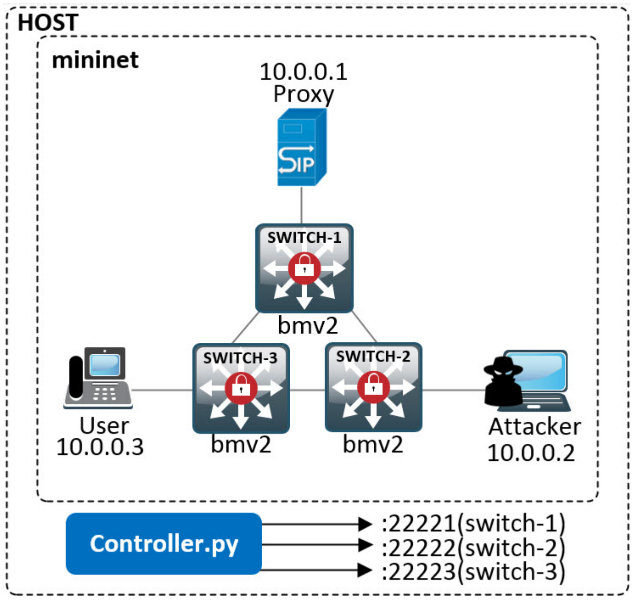


Fig. 2: Test environment

B. Workflow

The workflow consists of three processes which work together i.e. distributed sensors, centralized policy enforcement and attack mitigation. The steps for building each of these processes are described in the next section.

- **Distributed sensors**. Every port on P4-switch was designed to function as a sensor and capable of performing deep packet inspection at the application layer. For example, every time P4-switch received a packet, the headers (Ethernet, IP, UDP, SIP) were parsed and analyzed. With this approach, each port on P4-switch could count the number of SIP INVITE that were sent by the node connected to this port.
- **Centralized attack detection and policy enforcement**. A P4-controller was used to control these distributed sensors. It involved tasks such as initializing the sensors, evaluating sipinvite_counter every second, checking threshold for attack detection, and sending commands to P4-switch. For attack detection, the controller calculated the rate of SIP INVITE packets received by each port on P4-switch. When the rate exceeded the threshold, the controller sent a command to P4-switch to drop subsequent SIP INVITE packets coming from the same ingress port.
- **Distributed attack mitigation**. After drop command was received from P4-controller, P4-switch performed attack mitigation by dropping subsequent SIP INVITE packets that arrived at the offending port. For example, when the controller sent a command to P4-switch to drop subsequent SIP INVITE packets coming from port 2, P4-switch will only drop SIP INVITE packets that were coming from port 2. Please note about the use of port number (instead of source IP address) as drop criteria because the source IP address are generally spoofed during a DDoS attack. Also note that this mitigation only affected SIP INVITE packets and did not affect other packets e.g. ICMP, HTTP, etc.

C. Methods

Assumptions and baseline. The threshold used for SIP DoS attack was 10 SIP INVITE packets per second. The assumption is that human does not initiate more than ten calls within one second period. When the threshold exceeded, we assume that malware program was running on the computer and generated those calls. The attack was assumed to stop when the threshold was not exceeded for one consecutive minute. This period was selected so that the legitimate user would be able to make call as soon as the malware stopped.

To create a baseline and simulate normal working condition, the attacker host (10.0.0.2) was sending SIP INVITE packet to SIP Proxy once per second. Since it was less than 10 SIP INVITE per second, it did not trigger the defense mechanism, and P4-switch let the packets passed through. To simulate SIP DDoS attack, call rate was changed from sending 1 call to 1001 calls per second.

The focus of this experiment was not to stress test the system, but rather, to verify correct operation of SIP DDoS detection and mitigation capability.

Building distributed sensors. Distributed sensors were built by writing P4 code to run on *bmv2*. The code provided the switch with instructions for its data plane i.e. how to process each packet. It started off with packet type definition, followed by parsing, then ingress and egress pipeline, and finally, deparsing (Figure 3).

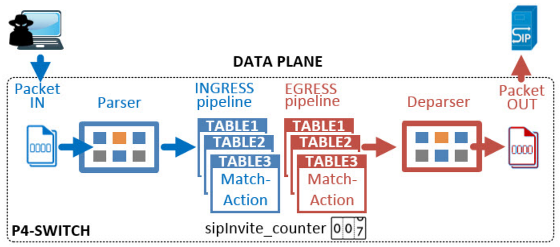


Fig. 3: Data plane programming defines how to process each packet

Packet type definition. Packet type was created by defining its headers, fields and field size in bytes. For example, Ethernet frame was characterized by three fields i.e. 48 bits of destination address, 48 bits of source address, and 16 bits of Ether type. IPv4, UDP, and SIP packets were also defined similarly, field by field. These fields would be used in the later stage as matching criteria for each packet that pass through the switch.

Parsing. The parser was used to extract packet headers and parser selection based on the hex value of a field that is evaluated. The selection will determine whether further recursion is required. For example, the parser extracted Ethernet header and looked at the value of `etherType` field. Since the value of this field was `0x0800` (hex value for IPv4), a parser called `parse_ipv4` was selected. In turn, `parse_ipv4` evaluated the value of `ipv4.protocol` field and found `0x11` (hex value for UDP), and a parser called `parse_udp` was selected. The same process applied for `parse_udp` where it evaluated `latest.dstPort` field and found `0x13C4` (hex value for 5060 in decimal i.e. SIP) and then `parse_sip` was selected. This recursive process continues until no more parsing is required and processing is passed to ingress pipeline.

Match-Action Table. A table was used to define the fields available for matching criteria, as well as the action required when there was a match. Besides defining what to match, the table also specified how to match e.g. ”LPM” for longest prefix match, ”range” for low and high values, ”index” for the index value of a table entry. SIP INVITE table was created to match SIP INVITE packets.

Ingress and Egress pipeline. Within ingress and egress pipeline, there were three tables: `ipv4_lpm`, `forward`, and `sipinvite_table`. The first two tables were used for normal IP packet forwarding, while the third was used to look for SIP INVITE. In this table definition, two match conditions were evaluated i.e. `ingress_port` and SIP INVITE. Each of this condition must be an exact match. When both was a match, there were two possible actions available i.e. `_drop` to drop the packets and `_nop` (no operation) to let the packets through.

Counter. A counter was defined to keep a record of the number of SIP INVITE packets received by each port. This counter was linked to `sipinvite_table` which means that for every row in that table, there would be a counter that keeps track of the number of packets that matched the criteria specified in `sipinvite_table`. In our example, we could check the number of SIP INVITE packets that were received from every port on the switch.

Combining sipinvite_table and sipinvite_counter definition resulted in a table like TABLE II. In this example, there were 24 ports on P4-switch, and there were 1452 SIP INVITE packets were received from port 2, 231 packets from port 3, and 0 from port 24.

Building centralized attack detection and policy enforcement. The controller established a control session to P4-switch to send commands. In chronological order, a controller performed three tasks: initialization, evaluating `sipinvite_counter` and making pass/drop decision.

Initialization. P4-controller initialized P4-switch by

Port number	INVITE	Counter
1	0x494e56495445	0
2	0x494e56495445	1452
3	0x494e56495445	231
–	–	–
24	0x494e56495445	0

TABLE II: sipinvite_table with sipinvite_counter.

inserting matching criteria and its corresponding action into match-action tables that were defined previously e.g. `table_add sipinvite_table _nop 2 0x494e56495445 =>`. This command created an entry in `sipinvite_table` with default action `_nop` for packets that met two criteria i.e. it came in at physical switch port `2` and for the first line of SIP packet, it contained the hex value of `0x494e56495445` i.e. ”INVITE”. When a match was found, `sipinvite_counter` was increased by 1.

Evaluating sipinvite_counter. Attack detection was achieved by checking `sipinvite_counter` on every switch port at every second. For example `"counter_read sipinvite_counter 2"` to check the number of SIP INVITE packets that had been received by port 2. The controller then calculated the delta between the value of the previous check against the value of the current check.

Making pass or drop decision. When the delta was greater than 10 (i.e. more than 10 SIP INVITE packets per second), SIP_DOS attack was detected, and DOS mode was activated for that port. When the delta was lower than the threshold for six consecutive checks i.e. one full minute, the mode was set to normal mode. One minute was chosen to strike a balance between blocking INVITE packets from DoS attack versus allowing user generated INVITE packets to go through. When the attack stopped, a legitimate user should be allowed to initiate a call in the next 60 seconds.

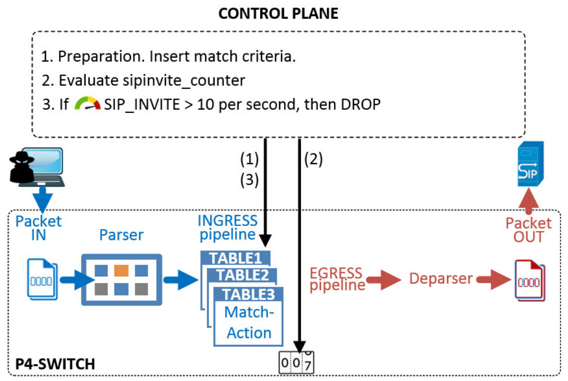


Fig. 4: Control plane defines what packet to match, checks counters, and sends commands to switch

Building distributed attack mitigation. When DOS mode was activated for a particular port, the P4-controller sent a command to the P4-switch to change the default action from ”pass” to ”drop.” e.g. `"table_modify sipinvite_table _drop 2"`. After this command had been executed, subsequent SIP INVITE packets that arrived at port 2 were dropped. To return to normal mode, the

P4-controller sent "`table_modify sipinvite_table _nop 2`" to P4-switch, and subsequent SIP INVITE packets received at port 2 were allowed to pass through as per normal.

Testing method. Attack simulation was done through changing the call-rate variable on SIPp. Chronologically, the test was done as follows:

- *Before the attack.* On the SIP-Proxy (10.0.0.1), SIPp was launched with "`sipp -sn uas -bg`". On the attacker host (10.0.0.2), SIPp was initially launched with "`sipp -sn uac 10.0.0.1 -r 1`" to start 1 new call every second. ICMP session was also launched to ping SIP-Proxy once every second. On the user host (10.0.0.3), SIPp was launched with "`sipp -sn uac 10.0.0.1 -r 1 -bg`" to start 1 new call every second.
- *Starting SIP DoS attack.* On the attacker host, the call-rate on SIPp was manually changed from 1 to 1001 by pressing "*" character. During the DoS attack, ICMP session should still be running as normal and not blocked. When the SIP-Proxy was still able to receive ICMP packet, it demonstrated that the filter only affects SIP packets. Other applications on the same host (e.g. ICMP) should still be allowed to pass through. On the user host, SIPp session should still be allowed to reach SIP-Proxy as per normal.
- *Stopping SIP DoS attack.* On the attacker host, the call-rate on SIPp was manually changed from 1001 to 1 by pressing "/" character.

Test verification. Two packet capture sessions were running, one at the switch port attached to the attacker (labeled as "A"), while the other running at the switch port attached to the SIP-Proxy (labeled as "B"). At point "A", we should see SIP and ICMP packets sent by the attacker host. At point "B", we should see SIP packets from the legitimate user and ICMP packets from the attacker. During normal condition, we should also see SIP packets from the attacker at point "B", whereas during DOS mode these SIP packets from the attacker would have been dropped by switch-2. This scenario is depicted in Figure. 5.

Evaluation and success criteria. There were three areas that were evaluated: detection and mitigation capability, time to detect and mitigate, and system resource consumption. Results are detailed in the next section.

The following variables were tracked during this experiment:

- The number of packets for SIP INVITE (sent & received). This number was taken from packet capture session at point A and B.
- The number of packets for ICMP (sent & received). This number was taken from packet capture session at point A and B.
- CPU & Memory consumption. This number was taken from `top` system utility running on the host system.

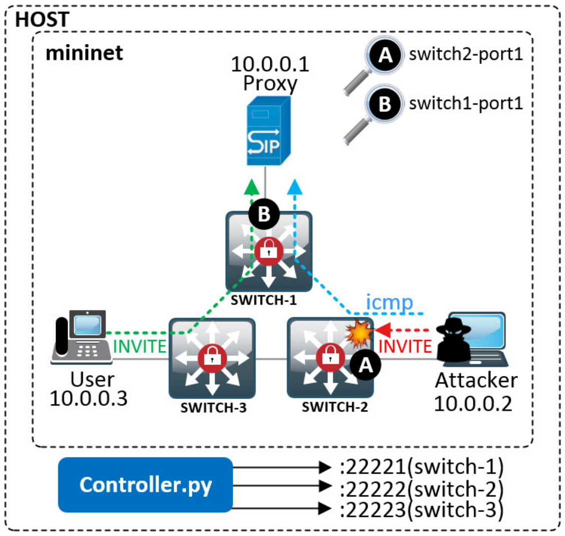


Fig. 5: Switch-2 dropped SIP INVITE packets while allowing ICMP packets to pass through. SIP-Proxy was still available to serve the legitimate user.

III. Results

In this section, we will see test results from three areas of evaluation: SIP DoS detection and mitigation, time to detect and mitigate, CPU and memory consumption.

A. *SIP DoS detection and mitigation*

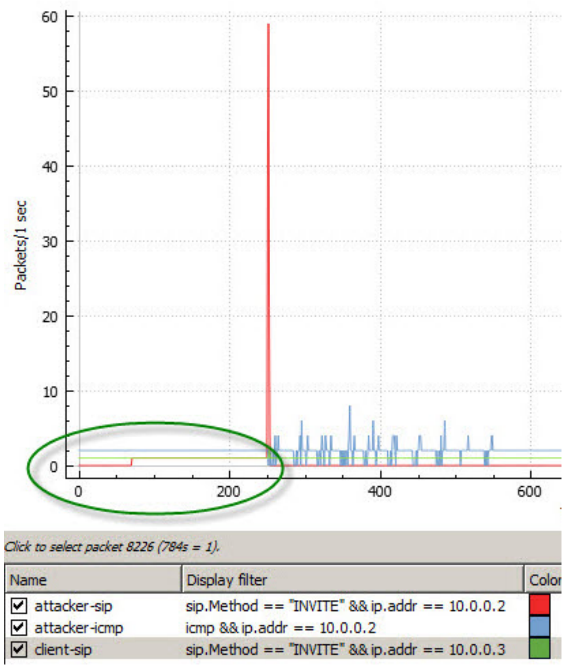
Before attack. Three sessions were observed at point B where the SIP-Proxy was connected (Figure 6). i.e. SIP packets from the attacker host (red line), ICMP packets from the attacker host (blue line), and SIP packets from the user host (green line). The X axis was the time (in seconds) since the session was started. The Y axis was the rate of SIP INVITE packet per second. We observed that the SIP-Proxy started receiving a burst of SIP INVITE packets (DoS attack) after 249 seconds since the session started.

During the attack. The call rate on the attacker host was increased from 1 call per second to 1001 calls which exceeded the threshold limit. Increased packets from the attacker host were observed at Point A where the attacker was connected (Figure. 7) . When P4-controller identified this condition as SIP DoS attack, it sent a command to switch-2 to start dropping SIP INVITE packets.

Observation from Point B (the SIP-Proxy side) shown an initial burst of SIP INVITE packets from the attacker host. Once switch-2 started dropping the packet, SIP-Proxy no longer received SIP INVITE from the attacker host (circled) as depicted in Figure. 8.

After the attack. Call rate on the attacker host was decreased from 1001 to 1 which was under attack threshold and therefore P4-switch sent a command to switch-2 to unblock the port.

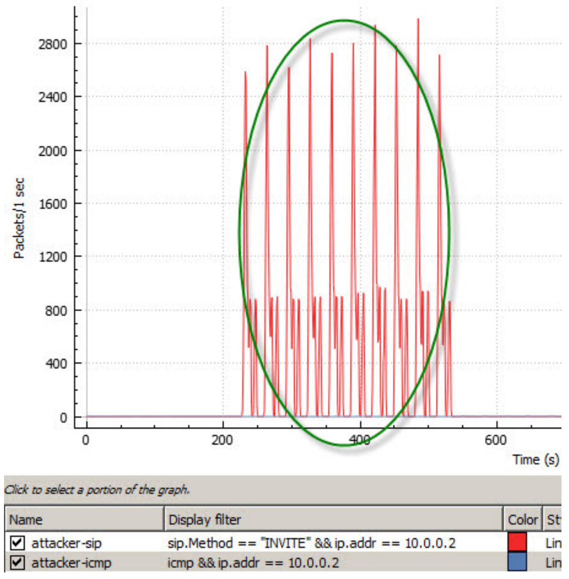
ICMP packets were not affected. When P4-switch dropped SIP INVITE packets, ICMP packets were still delivered as usual (as depicted by the blue line in Figure 8).



Click to select packet 8226 (784s = 1).

Name	Display filter	Color
✓ attacker-sip	sip.Method == "INVITE" && ip.addr == 10.0.0.2	🟥
✓ attacker-icmp	icmp && ip.addr == 10.0.0.2	🟦
✓ client-sip	sip.Method == "INVITE" && ip.addr == 10.0.0.3	🟩

Fig. 6: Before attack period (circled): SIP-Proxy received SIP & ICMP packets from the attacker host.



Click to select a portion of the graph.

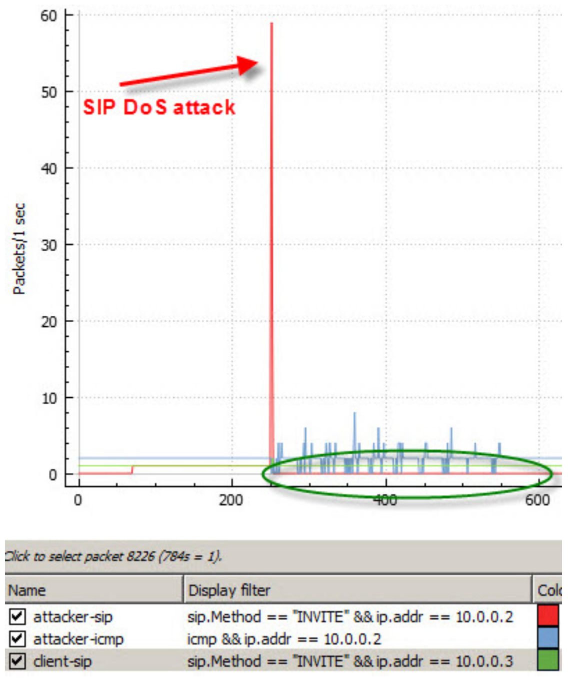
Name	Display filter	Color	St
✓ attacker-sip	sip.Method == "INVITE" && ip.addr == 10.0.0.2	🟥	Lin
✓ attacker-icmp	icmp && ip.addr == 10.0.0.2	🟦	Lin

Fig. 7: Increased SIP INVITE packets from the attacker host.

This demonstrates that the filtering only affected SIP INVITE and not other packets. In the case where a computer was infected with botnet/malware, the user would still be able to use other applications while SIP INVITE packets generated by the malware are blocked.

B. *Time to detect and mitigate*

The time it took to start blocking SIP INVITE packets was 1.47 seconds. This period was calculated by subtracting



Click to select packet 8226 (784s = 1).

Name	Display filter	Col
✓ attacker-sip	sip.Method == "INVITE" && ip.addr == 10.0.0.2	🟥
✓ attacker-icmp	icmp && ip.addr == 10.0.0.2	🟦
✓ client-sip	sip.Method == "INVITE" && ip.addr == 10.0.0.3	🟩

Fig. 8: After initial burst of SIP INVITE, the SIP-Proxy is no longer received packets from the attacker (circled).

the time when the P4-controller sent the command from the last SIP INVITE packet received by the SIP-Proxy (Table. III).

Time the controller sent command to SW2 (in second from start)	Time the last INVITE packet received by SIP-Proxy (in second from start)	Time to block (second)
251.014599	252.162154	1.147555

TABLE III: Time taken to start blocking was 1.47 second.

C. *CPU and Memory resource consumption*

On Switch 1 (connected to the SIP-Proxy), there was a brief period of time where the CPU spiked to 24.8%. This was the time when the attack started. After mitigated, CPU consumption returned to normal. Consumption on the memory was spiked to 3% as depicted in Figure. 9

On Switch 2 (connected to the attacker), CPU consumption was peaked at 71% during SIP DoS attack, but on average it was at 24.49%. (Figure 10). Memory consumption was peaked at 2.9%.

IV. Discussion

In contrast to the destination-based defense strategy that is commonly deployed to deal with SIP DDoS attack, this paper introduced source-based defense strategy i.e. attack detection and mitigation is performed at the location where the attack is originated. This proposal utilizes P4 for SIP DoS attack detection and SDN controller for attack mitigation. P4 offers rich programmability and metadata. Coupled with control plane, it provides centralized security policy management with the capability to enforce the policy on many remote switches.

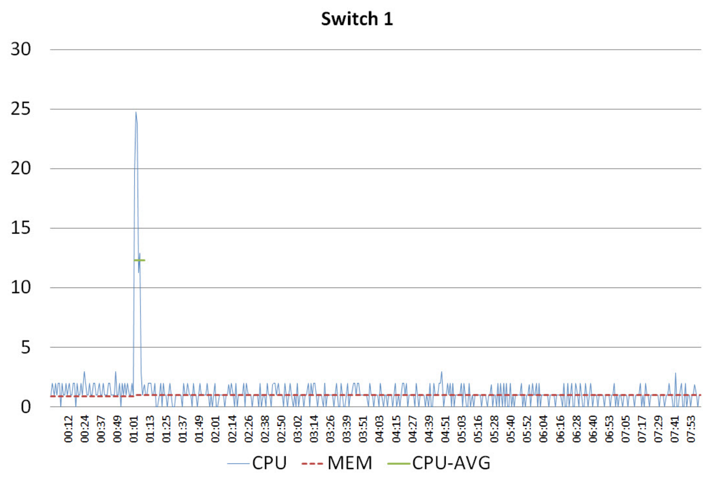


Fig. 9: CPU and Memory consumption on Switch 1 (attached to the SIP-Proxy)

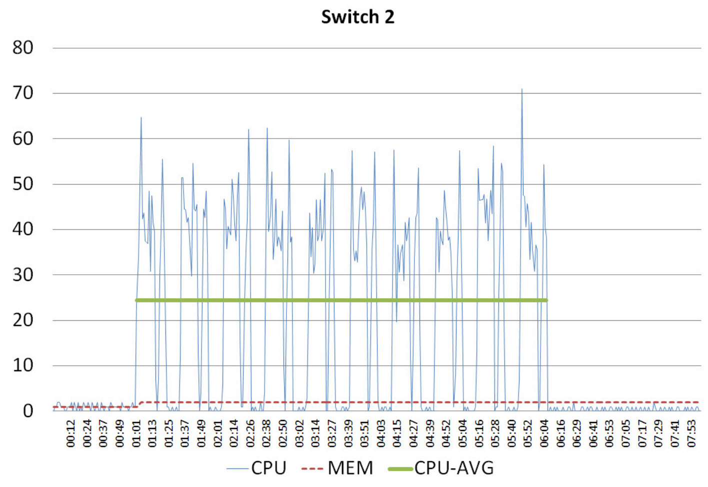


Fig. 10: CPU and Memory consumption on Switch 2 (attached to the attacker)

These results suggest that source-based approach worth serious consideration for deployment.

The result highlights that source-based defense was able to deliver effective attack detection and mitigation. It did that at the first network hop, and therefore attack-related traffic did not propagate further upstream. Furthermore, the tasks were accomplished within reasonable system resource consumption. With SIP INVITE generated at 1001 calls per second, it took 1.47 seconds to detect and mitigate the attack. During the attack, average CPU utilization was 24.49% (peak at 71%). Memory consumption was peaked at 3%.

The result also displayed that while in blocking mode, it did not affect other legitimate traffic as demonstrated by ICMP packets. Ping request and reply packets were still able to flow through while SIP DDoS was in blocking mode.

The authors [3] states that ”An ideal comprehensive DDoS defense mechanism must have specific features to combat DDoS flooding attacks both in real-time and as close as possible to the attack sources.” To that end, the result presented in the previous section met this description.

For high-speed performance requirements, hardware-accelerated chip are available i.e. instead of running P4 code on software or on general purpose CPU; it can also be executed on NPU (Network Processor), FPGA (field-programmable gate array), or even ASIC (application-specific integrated circuit) [11][13]. Project whippersnapper provides P4 benchmark suite to compare the performance of various hardware platform [14].

Current limitation. At the time of writing, support for encrypted packets (e.g. TLS and DTLS) are not natively supported by P4 and therefore relies on functions provided by the underlying hardware.

V. Conclusion

This paper proposed First-hop SIP DDoS detection and mitigation using data plane programming (P4) and control plane programming of an Ethernet switch. This approach creates source-based defense component that can be combined with existing destination-based defense component to create more scalable SIP DDoS solution.

Aggressive deployment of Software-Defined Networking (SDN) and Network Function Virtualization (NFV) creates synergy and opportunity for this approach to be implemented organization-wide. Many organizations are upgrading their existing network infrastructure to support SDN and NFV concepts. They need it to support the requirement to innovate faster than their competitors and to grow their business. Since the concepts presented in this proposal are aligned with SDN and NFV concepts, this solution could be deployed on the same SDN and NFV infrastructure that they are using. This infrastructure-sharing concept alleviates the need for specialized hardware/software dedicated for this purpose and therefore ease the deployment effort.

The network operator has the option to plan for an incremental deployment. They could program P4-switch and SDN controller to perform the same functions as existing switches e.g. MPLS, VLAN, Routing, etc. The users would not experience service degradation or even aware that the underlying infrastructure has changed. Since P4-switch is backward-compatible with existing switch, it gives network operator with flexibility as for where and when they will deploy P4-capable switch.

In addition to incremental deployment, network operator also has the options to install P4-switch at their network edge e.g. in network-to-network or ISP-to-ISP interconnection scenarios. The ”First-hop” in this instance is not coming from an end-user, but rather from a network peer. P4-switch can provide SIP DDoS protection for traffic coming from a particular peer-network or ISP. Having this tool in place would motivate network operation team for deployment.

Another motivation for network operator would come in the form of financial benefits. In some cases, network operator would have to pay a fee to their peers for bandwidth usage (i.e. settlement fee) regardless whether it was legitimate traffic or traffic generated by a DDoS attack. As the operator can drop DDoS traffic at the edge of their network, they would not have to bear the cost of transporting DDoS traffic. This financial benefit would provide objective and concrete business justification for the deployment.

References

[1] ”NG911 Cybersecurity Primer FINAL 508C (003).pdf”, Department of Homeland Security, 2017. [Online]. Available: https://www.dhs.gov/sites/default/files/publications/NG911%20Cybersecurity%20Primer%20FINAL%20508C%20(003).pdf. [Accessed: 09- Jun- 2017].

[2] J. Yu, ”An Empirical Study of Denial of Service (DoS) against VoIP,” 2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS), Granada, 2016

[3] S. T. Zargar, J. Joshi, and D. Tipper, A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks, IEEE Commun. Surv. Tutorials, vol. 15, no. 4, 2013.

[4] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker, ”P4: Programming protocol-independent packet processors”, SIGCOMM Comput. Commun. Rev., vol. 44, no. 3, pp. 87-95, Jul. 2014.

[5] G. Ormazabal, S. Nagpal, E. Yardeni, and H. Schulzrinne, Secure SIP: A scalable prevention mechanism for DoS attacks on SIP based VoIP systems, in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2008.

[6] F. Huici, S. Niccolini, and N. DHeureuse, Protecting SIP against very large flooding DoS attacks, GLOBECOM - IEEE Glob. Telecommun. Conf., 2009.

[7] I. Hussain, S. Djahel, Z. Zhang and F. Nat-Abdesselam, ”A comprehensive study of flooding attack consequences and countermeasures in Session Initiation Protocol (SIP)”, Security Comm. Networks, 2015

[8] Z. Liu, X. Yin and H. Lee, ”An Efficient Defense Scheme against SIP DoS Attack in SDN Using Cloud SFW,” 2014 Ninth Asia Joint Conference on Information Security, Wuhan, 2014

[9] B. Astuto, a Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks, Ieee Commun. Surv. Tutorials, vol. 16, no. 3, 2014.

[10] A. D. Keromytis, ”A Comprehensive Survey of Voice over IP Security Research,” in IEEE Communications Surveys & Tutorials, vol. 14, no. 2, Second Quarter 2012.

[11] P. Bencek, V. Pu, H. Kubtov, ”P4-to-VHDL: Automatic Generation of 100 Gbps Packet Parsers”, 2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), May 2016.

[12] Q. Yan, R. Yu, Q. Gong, and J. Li, Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges, IEEE Commun. Surv. Tutorials, no. c, 2015.

[13] P. Bosshart, G. Gibb, H. Kim, G. Varghese, N. Mckeown, M. Izzard, F. Mujica, and M. Horowitz, Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN, Acm Sigcomm, 2013.

[14] H. T. Dang, H. Wang, T. Jepsen, G. Brebner, J. Rexford, R. Soul, and H. Weatherspoon, Whippersnapper : A P4 Language Benchmark Suite.

[15] p4lang, p4lang/behavioral-model, GitHub, 07-Jun-2017. [Online]. Available: https://github.com/p4lang/behavioral-model. [Accessed: 12-Jun-2017].

[16] p4lang, p4lang/tutorials, GitHub, 01-Jun-2017. [Online]. Available: https://github.com/p4lang/tutorials. [Accessed: 12-Jun-2017].

[17] p4lang, p4lang/p4c-bm, GitHub, 09-May-2017. [Online]. Available: https://github.com/p4lang/p4c-bm. [Accessed: 12-Jun-2017].

[18] M. Team, Mininet, Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet. [Online]. Available: http://www.mininet.org/. [Accessed: 12-Jun-2017].