SDN-based Network Security Functions for Effective DDoS Attack Mitigation

Daeyoung Hyun*, Jinyoug Kim†, Dongjin Hong†, and Jaehoon (Paul) Jeong‡
* Department of Software, Sungkyunkwan University, Republic of Korea
† Department of Computer Science & Engineering, Sungkyunkwan University, Republic of Korea
‡ Department of Interaction Science, Sungkyunkwan University, Republic of Korea
Email: {dyhyun, timkim, dong.jin, pauljeong}@skku.edu

***Abstract*—Distributed Denial of Service (DDoS) attack has been bringing serious security concerns on banks, finance incorporation, public institutions, and data centers. Also, the emerging wave of Internet of Things (IoT) raises new concerns on the smart devices. Software Defined Networking (SDN) and Network Functions Virtualization (NFV) have provided a new paradigm for network security. In this paper, we propose a new method to efficiently prevent DDoS attacks, based on a SDN/NFV framework. To resolve the problem that normal packets are blocked due to the inspection on suspicious packets, we developed a threshold-based method that provides a client with an efficient, fast DDoS attack mitigation. In addition, we use open source code to develop the security functions in order to implement our solution for SDN-based network security functions. The source code is based on NETCONF protocol [1] and YANG Data Model [2].**

***Keywords*—Software Defined Network, Netconf & YANG, Distributed Denial of Service, Network Function Virtual, Suricata.**

I. Introduction

Internet services of a client can be destroyed seriously by a Distributed Denial of Service (DDoS) attack that attempts to flood packets to the client servers. According to Internet of Things (IoT) and cloud services, the network environment can have numerous traffic in the future. The traffic of ordinary users can also be exploited by DDoS attacks. Because of its distributed feature and diverse sources, a client can hardly predict a DDoS attack. Many researchers and network security companies have been investigating and developing effective methods to prevent, mitigate, and detect the DDoS attack. Those methods include blackhole routing, intrusion prevention systems (IPS), DDoS based defence, etc. Although those methods can alleviate the attack, the services of a client may still be affected considerably. Another drawback of those methods is the flexibility issue. The DDoS attack are becoming more sophisticated than before, this requires that a defence method shall quickly adapt a new DDoS attack pattern.

As alternatives, Software Defined Networking (SDN) and Network Function Virtualization (NFV) are emerging technologies for next genergeation networks. The SDN and NFV are designed for a configuration-free network, which has a lot of efficiency such as operation cost reduction and vendor independency. In particular, the Internet Engineering Task Force (IETF) Interface to Network Security Functions (I2NSF) Working Group has proposed and architecture based on SDN and NFV. SDN refers to a technology that can flexibly control and configure network paths and traffic by configuring an existing network composed of hardware as software. NFV suggests a paradigm that various hardware technologies constituting the software are converted into software and provides as a service [3]–[7]. Currently, SDN and NFV technologies are being studied in fusion. The reason is to make effective use of various NFV technologies in SDN where flexible network configuration is possible.

In this paper, we propose an architecture based on the I2NSF framework opensource to mitigate DDoS attack. For the prevention of a DDoS attack, web server administrator sets preemptive security rule. When a DDoS attack is finished, it will supplement and reinforce additional security rules based on the result of the attack. In addition, it is difficult to distinguish general packets or DDoS attack packet. All packets go through an SDN controller to apply DDoS attack mitigation proposed in this paper. It allows the packet to be accommodated with a flexible probability set by the server administrator, and the DDoS attack as well as the general packet in which traffic increases at a specific time.

We use several opensource codes to develop the network configuration functions. The new developed network functions are open to the community in order to receive improvement. Especially, in this paper, Suricata [8] is used to configure OpenDaylight [9], Intrusion Detection System (IDS) in Mininet [10] and SDN controllers, and Netconf & YANG [1], [2] is used as a configuration management protocol and modeling language.

The composition of this paper is as follows. Section II describes the principles and suricata of DDoS and the description and background of Netconf & YANG, Section III describes the improved architecture proposed by the IETF I2NSF Working Group. Section IV shows the implementation of DDoS attack against DDoS attacks based on the proposed I2NSF framework. Section V, future research directions are presented based on the results of the research. Finally, Section VI summarizes the results.

II. Overview

This section describes the types of Distributed Denial of Service (DDoS) attacks and techniques used to implement and prevent DDoS attacks in Software Defined Networking (SDN)

and Network Function Virtualization (NFV) introduction. In particular, Network Configuration (Netconf) Protocol & Yet Another Next Generation (YANG) [1], [2] used to send data received from web server manager to SDN controller, threshold setting method for using security rule in surciata with Intrusion Detection System (IDS) function by opensource, type of DDoS attack, it deals with contents.

A. Distributed Denial of Service (DDoS) Attack

- Defined of DDoS Attack : A distributed denial of service attack is an attack method that paralyzes servers by instantly acquiring network bandwidth by abnormally increasing traffic by accessing a specific web site by a large number of computers on the internet or a network. At this time, the server has a function of distributing traffic, but it is called "Distributed Denial of Service attack" in the sense of neutralizing the traffic.

- Purpose of DDoS Attack : The purpose of a DDoS attack is to prevent the target server from receiving a legitimate signal. In order to attack a specific site, a hacker puts tools for attacking the service on several computers and causes a huge number of packets to be simultaneously generated, causing a paralysis of the server system. As a result, users will not be able to use the server.

- Attack Method of DDoS Attack : Step1, malicious hacker injects malicious code into an unspecified number of personal computers via email or file transfer. Step2, when you open an e-mail containing a malicious code or execute a file, the computer becomes botnet. Step3, the user then uses the computer without knowing the above. Step4, here, malicious code spreads to more users. Step5, then, according to the attack command of the hacker to give a lot of traffic to a specific target service to paralyze the service.

- Response Method of DDoS Attack : There are two ways to respond to DDoS attacks that evolve over time. It is divided into a way to put the function in the existing network environment and a method to construct the new network mirroring environment.

1) In-Line : It is a method to detect and block attacks by monitoring the traffic coming into the server in real time by placing a function corresponding to the DDoS attack in front of the firewall. In terms of processing in real time, security is high, but there is bandwidth and load of network line.
2) Out-of-Path : In the current network configuration, a function to cope with a DDoS attack is separately configured to replicate traffic to the server, analyze the traffic, and then block the traffic. Although the bandwidth and load of the network line is small, the response is delayed compared to the real-time processing, and the construction cost is high.

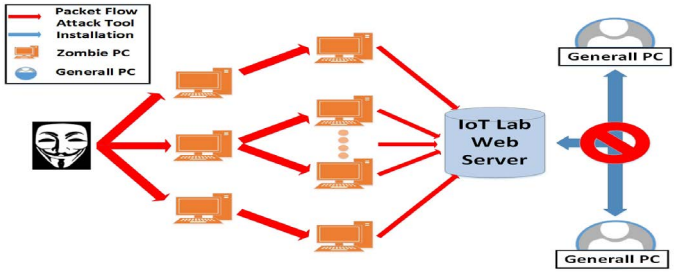

Fig. 1. Typical DDoS Attacks through Botnet

B. Suricata

Suricata is an opensource that acts as a representative Intrusion Detection System (IDS) & Intrusion prevention systems (IPS). It has features such as IDS & IPS, high performance, automatic protocol detection, network security monitoring, rua scripting and industry standard outputs. Various security policies are provided by default, and the policies are automatically updated regularly. In particular, security experts establish security rules in consideration of security incidents that occur in the meantime and future security incidents. Snort's security policies are also compatible with previous versions of suricata, providing even more flexibility. The reason is that suricata is an opensource created by improving Snort. Suricata consists of packet type, source, destination, and rule content. The actual rule is defined as below [8].

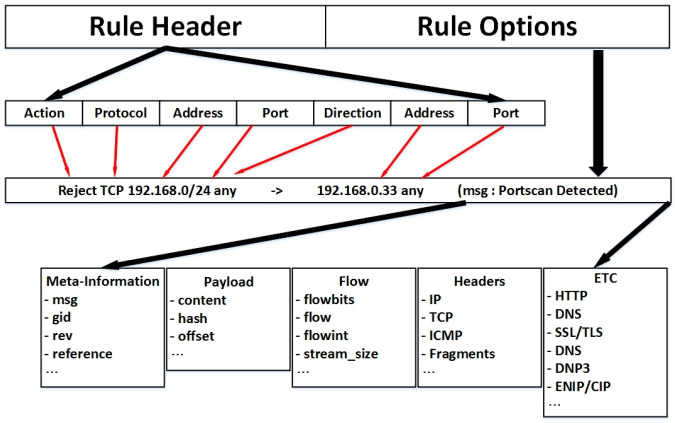

Fig. 2. Security Policy Analysis of Suricata

Suricata security rules are divided into rule header and rule options. The rule header consists of action, protocol, src-address, src-port, direction, dest-address, and dest-port. Rule options include additional information about the rule and other information.

C. Netconf & YANG

Netconf & YANG [1], [2] is a protocol for network configuration management and a modeling language for it. The relationship between the two elements invokes network

components through the Netconf and determines the format of the events that are invoked for network configuration. YANG contains built-in data type information or specific data type information for the application. Tips for utilizing Netconf & YANG, are that it is easy to add / delete / update components in the future after the data model is completed, since accurate and flexible structure design is required at the information model stage.

```
module : ietf-i2nsf-nsf-facing-interface
+--rw cfg-network-security-control
|    +--rw policy
|         +--rw policy-name string
|         +--rw policy-id string
|         +--rw rules* [rule-id]
|              +--rw rule-name string
|              +--rw rule-id uint 8
|              +--rw rule-msg string
|              +--rw rule-rev uint 8
|              +--rw rule-gid uint 8
|              +--rw rule-class-type string
|              +--rw rule-reference string
|              +--rw rule-priority uint 8
|              +--rw event
|              |  +--rw user-security-event* [usr-sec-event-id]
|              |  |    +rw usr-sec-event-id uint 8
|              |  |    +rw usr-sec-event-content string
|              |  |    +rw usr-sec-event-format uint 8
|              |  |    +rw usr-sec-event-type uint 8
|              |  +--rw device-security-event* [dev-sec-event-id]
|              |  |    +rw dev-sec-event-id uint 8
|              |  |    +rw dev-sec-event-content string
|              |  |    +rw dev-sec-event-format uint 8
|              |  |    +rw dev-sec-event-type uint 8
|              |  |    +rw dev-sec-event-type-severity uint 8
|              |  +--rw system-security-event* [sys-sec-event-id]
|              |  |    +rw sys-sec-event-id uint 8
|              |  |    +rw sys-sec-event-content string
|              |  |    +rw sys-sec-event-format uint 8
|              |  |    +rw sys-sec-event-type uint 8
|              |  +--rw time-security-event* [time-sec-event-id]
|              |  |    +rw time-sec-event-id uint 8
```

Fig. 3. Information Model for DDoS Attacks Mitigation

```
leaf policy-name {
      type string;
      mandatory true;
      description
            "The name of the policy.
             This must be unique.";
}
leaf policy-id {
      type string;
      mandatory true;
      description
            "The ID of the policy.
             This must be unique.";
}
list rule {
      key "rule-id";
      description
            "This is a rule for network security control.";

      leaf rule-name {
      mandatory true;
      description
            "The name of the rule.
             This must be unique.";
      }
      leaf rule-id {
            type uint8;
            mandatory true;
            description
                  "The Id of the rule.
                   This is key for rule-list.
                   This must be unique.";
      }
}
```

Fig. 4. Data Model for DDoS Attacks Mitigation

The following is the process of outputting desired data using Netconf & YANG.

- Create Information Model Structure : Specify each component and data type and create the basic skeleton of YANG.

- Create Data Model : Based on the information model, specify the data type and the necessity of the element together with a detailed description of each component. When reading actual data, the data type and the necessity of element are reflected.

- Use Data : Through the above steps, you can change the high-level data to low-level data. It means that the administrator and the user can utilize it according to the intended use. In this paper, we try to provide stable web service by controlling the flow of normal packets and suspicious packets by using data.

- XML Encoding : In this step, the administrator and the user set a desired security rule in step 3, and then prepare data for transmission to the NSF through the SDN controller after modifying the data in XML form.

III. Architecture of I2NSF

In this section, we present the framework of the IETF I2NSF Working Group and describe the framework underlying the implementation phase of this paper. In the first proposed I2NSF framework, security functions are composed of suricata [8] which is an opensource [11]–[15].

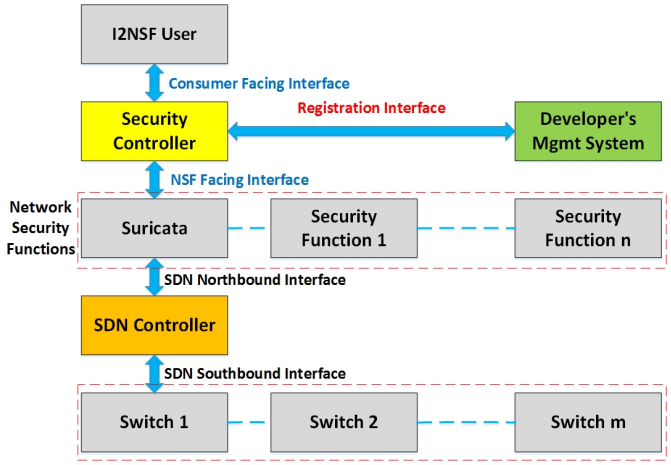

Fig. 5. I2NSF Architecture with DDoS Attack Mitigation

A. Flow of Architecture

1) The client delivers the high-level security policy desired by the administrator to the security controller.
2) The security controller translates the received high-level security policy into a low-level security policy and delivers it to the security function. Netconf [1], a network configuration protocol, and YANG [2], a data modeling language, are used for translation.
3) Security functions with translated security policy are forwarded to each switch through SDN Controller. This paper intends to use opensource as a security function.

4) It allows the security administrator's policy to be applied to each host. The framework can be divided into consumer facing interface, NSF facing interface, and registration interface [15].

B. Architecture Description

- The consumer facing interface is a security service manager that delivers a high-level security policy to the security controller through client web / android device.
- The NSF Facing Interface translates the received high-level security policy into low-level for execution in the NSFs. The security controller passes the requested NSF Facing Interface to the appropriate NSFs to execute the requested security service.
- The registration interface can add functions to NSFs using security controller [15].

IV. Implementation

This paper is based on the scenario that a Distributed Denial of Service (DDoS) attacks by generating a lot of packets. A number of botnets infected by a malicious hacker generates a packet targeting a web server that is an attacking server. However, the web server that built the defense system against the DDoS attack already bypasses the traffic arrived from the botnet to suricata of the Network Security Function (NSF) for the DDoS attack in the SDN controller. Here suricata blocks the traffic according to the packet threshold set by the web server administrator.

A. Flow Chart of I2NSF Architecture

We use suricata, which has IDS function, to control the flow of normal and unidentified packets for DDoS attack mitigation. In this case, the control is to set and limit the packet flow threshold in a range that enables smooth service in the destination server (web server). The network administrator applies the security rule through the I2NSF architecture. The process is as follows.

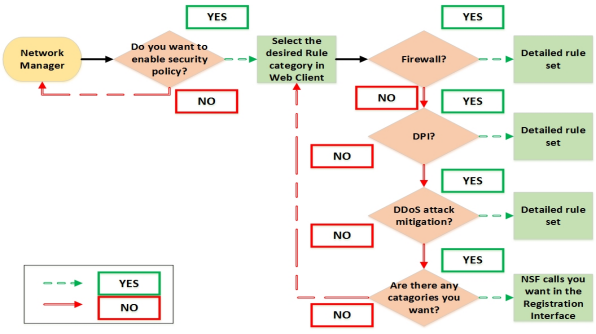

Fig. 6. Flowchart for an I2NSF Architecture when Establishing a Security Policy

If the network administrator wants to apply security policy, you need to connect to the web client. Then select the required NSF in the web client and set detailed rules. The network administrator can also use multiple NSFs rather than a single NSF. In this experiment, DDoS attack mitigation is used.

B. Block According to Threshold Setting

In the web client where the administrator can set the rule easily, set the rule to respond to the DDoS attack. Rule setting in the form of "reject ip srcIP any - destIP any (msg: "IoTLab Website Reject"; threshold: type threshold, track by-src, count 2, seconds 10; sid: 118; rev: 1;)". The botnet generates a packet at the homepage "http://iotlab.skku.edu/". However, the homepage administrator sets a threshold value for the traffic for the stable operation of the homepage by using the web client connected to the SDN.

C. Network Topology

The network topology is divided into A, B, C, and D. Section A consists of general users and botnets for DDoS attack. This assumes that a hacker is infected with a bonet by an ordinary user. Section B consists of a firewall, Deep Packet Inspection (DPI), and IDS. This experiment uses NSF using Suricata. Section C consists of the Service Function Forwarder (SFF). In order to use various NSFs dynamically in the future. Section D is a section for administrators. The administrator configures this section to establish the security policy. The following figure shows the network topology used in the experiment [9], [10].

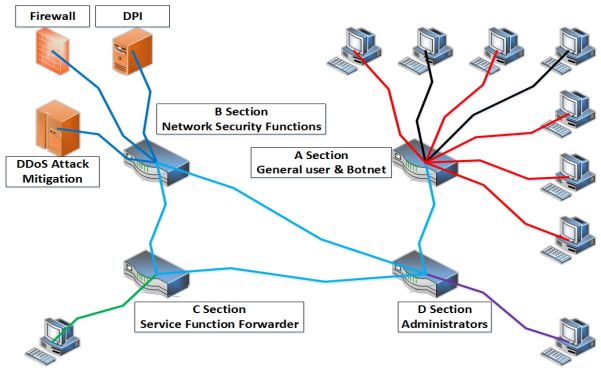

Fig. 7. Experimental Topology for DDoS Attack Mitigation

D. Experimental Method

In this experiment, if more than 1, 3, 5, 7, and 9 packets are received within 10 seconds, the packet is blocked. That is, we examine the amount of packet traffic arriving at the same time and examine the relationship between the threshold and the packet drop rate. Also, we aim to provide stable and consistent service by adjusting the amount of packet traffic without distinction between normal user and botnet.

E. Implementation Result

Experiments show that the packet drop rate is changed by the threshold that limits the amount of packet traffic allowed per second. More specifically, it was confirmed that amount of packets received at a desired level can be actually adjusted by simultaneously transmitting packets to the same destination and setting different packet threshold values allowed for each bot on the same target. In particular, compared with the "Normal" which has no configuration, the total packet drop rate decreases as the packet drop rate of each bot with the individual policy set increases. The result is shown in Fig9. The light blue background means "Packet Count" which is different from each other, and the yellow background means "Packet Drop Rate" according to "Packet Count".

	Sent Packet	Packet Receive	Packet Drop Rate
bot1	71531	0	100.0000%
bot2	71535	50045	30.04124%
bot3	71536	57312	19.88369%
bot4	71536	64373	10.01314%
bot5	71535	64475	9.86929%
Normal	71535	71535	0%

Fig. 8. DDoS Attack Mitigation Result through Security Policy-Threshold

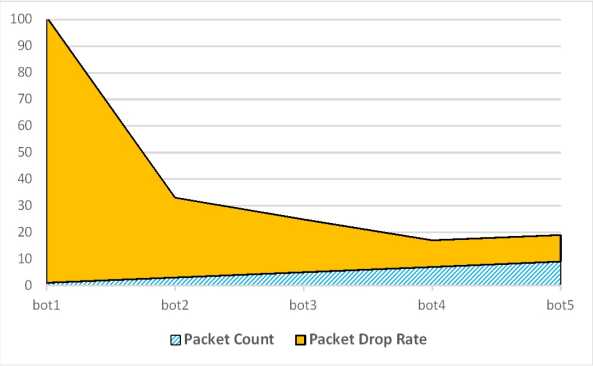

Fig. 9. Relationship between Packet Count and Packet Drop Rate

V. RESEARCH ISSUES

1) It should be open source compatible with various languages. Standardization or rules for this should be preceded. For example, it will use existing programming languages and new programming languages such as C, Python, Node.js, and javascript.
2) Currently, there are many studies on Software Defined Networking (SDN) controllers and Network Security Functions (NSFs). However, as more research progresses, it will be difficult to manage and certify newly developed NSFs. Therefore, there is a need for an efficient platform for management and authentication of many NSFs.
3) Research on the performance of SDN Controller is needed. Currently, only the architecture of a single SDN Controller has been considered. In the future, it is necessary to study the performance of each SDN Controller and the use of multiple SDN Controllers.
4) The research to date has been made using Mininet to mimic the actual network environment. In the future, openstack can provide SDN and NFV services in cloud computing services. This is under study for the year 2018.
5) This study aims at providing stable service of server by controlling the number of packets coming into the server per second without distinguishing between normal user and botnet. However, in the future, using packet analyzed in connection with DPI. We will implement a service that controls the packet flow by separating users.

VI. CONCLUSION

We have studied effective defense measures against Distributed Denial of Service (DDoS) attacks that have continued from the past. In order to utilize Software Defined Networking (SDN) to improve various inefficiencies such as network inefficiency and high cost, I2NSF architecture proposed by IETF's I2NSF working group was utilized to utilize the efficiency of network management. Administrators allows to configure NSF flexibly using opensource, which is an active field. This paper confirms that network administrators and web administrators can control the network packet capacity through the SDN controller according to the general environment of the company and the specific network environment. We will continue to apply this technology to the cloud-based openstack for real-world use. We will build a dynamic implementation base of the SDN controller through Deep Packet Inspection (DPI) with the firewall we have implemented. It will also tie the ability to distribute packets across various NSFs through load balancing.

ACKNOWLEDGMENT

This research was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (Ministry of Science and ICT, MSIT) (No.2016-0-00078, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning). This research was supported in part by the MSIT under the ITRC (Information Technology Research Center) support program (IITP-2017-2017-0-01633) supervised by the IITP. This research was also supported by the MSIP(Ministry of Science, ICT & Future Planning), Korea, under the "Employment Contract based Master's Degree Program for Information Security" supervised by the KISA(Korea Internet & Security Agency)(H2101-16-1001). Note that Jaehoon (Paul) Jeong is the corresponding author.

REFERENCES

[1] R. Enns, M. Bjorklund, J. Schoenwaelder and A.Bierman "Network Configuration Protocol (NETCONF)", IETF RFC 6241, June 2011

[2] M. Bjorklud “YANG-A Data Modeling Language for the Network Configuration Protocol (NETCONF)”, IETF RFC 6020, October 2010

[3] Recommendation ITU-T Y.3300, “Framework of Software-Defined Networking,” ITU-T, Jun 2014

[4] Open Networking Foundation, “SDN Architecture,” ONF, Jun 2014

[5] H. Kim and N. Feamster, “Improving Network Management, with Software Defined Networking,” IEEE Communications Magazine, vol. 51, no. 2, pp. 114-119, Feb 2013

[6] Open Networking Foundation, “OpenFlow Switch Specification (Version 1.3.0),” ONF, Oct 2013

[7] J. Jeong, H. Kim, and J. Park, “A Framework for Security Services based on Software-Defined Networking,” DC2, Mar 2015

[8] Suricata, “Opensource of IDS & IPS role”, accessed 2017. [Online]. Available: https://suricata-ids.org/

[9] OpenDaylight, “SDN controller”, accessed 2017. [Online]. Available: https://www.opendaylight.org/

[10] Mininet, “An instant virtual network on your laptop”, accessed 2017. [Online]. Available: http://mininet.org/

[11] D. Lopez, E. Lopez, L.Dunbar, J. strassner, and R. Kumar “Framework for Interface to Network Security Functions”, IETF draft-ietf-i2nsf-framework-05, May 2017

[12] S. Hares, R. Moskowitz, L.xia, J. Jeong, and J.kim “I2NSF Capability YANG Data Model”, IETF draft-hares-i2nsf-capability-data-model-01, March 2017

[13] J. Jeong, D. Daghmehchi, T. Ahn, R. Kumar, and S. Hares “Consumer-Facing Interface YANG Data Model for Interface to Network Security Functions”, IETF draft-jeong-i2nsf-consumer-facing-interface-dm-01, March 2017

[14] J. Kim, J. Jeong, J. Park, S. Hares, and L. Xia “I2NSF Network Security Functions Facing Interface YANG Data Model”, IETF draft-kim-i2nsf-nsf-facing-interface-data-model-01, March 2017

[15] J. Kim, M. Daghmehcih Firoozjaei, J. Jeong, K. Hyoungshick, and J. Park “SDN-based Security Services using Interface to Network Security Functions”, The 6th International Conference on ICT Convergence (ICTC) 2015, August 2015