Journal of Korea Multimedia Society Vol. 17, No. 8, August 2014(pp. 988-994)
http://dx.doi.org/10.9717/kmms.2014.17.8.988

Utilizing OpenFlow and sFlow to Detect and Mitigate SYN Flooding Attack

Muhammad Nugraha[†], Isyana Paramita[††], Ardiansyah Musa[†††], Deokjai Choi[††††], Buseung Cho[†††††]

ABSTRACT

Software Defined Network (SDN) is a new technology in computer network area which enables user to centralize control plane. The security issue is important in computer network to protect system from attackers. SYN flooding attack is one of Distributed Denial of Service attack methods which are popular to degrade availability of targeted service on Internet. There are many methods to protect system from attackers, i.e. firewall and IDS. Even though firewall is designed to protect network system, but it cannot mitigate DDoS attack well because it is not designed to do so. To improve performance of DDOS mitigation we utilize another mechanism by using SDN technology such as OpenFlow and sFlow. The methodology of sFlow to detect attacker is by capturing and sum cumulative traffic from each agent to send to sFlow collector to analyze. When sFlow collector detect some traffics as attacker, OpenFlow controller will modify the rule in OpenFlow table to mitigate attacks by blocking attack traffic. Hence, by combining sum cumulative traffic use sFlow and blocking traffic use OpenFlow we can detect and mitigate SYN flooding attack quickly and cheaply.

Key words: Firewall, SYN Flooding, sFlow, OpenFlow

1. INTRODUCTION

SYN flooding attack is one of the most popular Distributed Denial of Service attack methods to degrade availability of targeted service on Internet [1]. In normal connection, to communicate between host and server, they establish TCP connection with triple exchange of packets known as three-way handshake[2]: host sends SYN segment to server, and then server responds with SYN/ACK segment, and after that host responds with ACK segment (see Fig. 1). SYN flood attack inundates a site with SYN segments containing forged (spoofed) IP source addresses with nonexistent or unreachable addresses. Server responds with SYN/ACK segments to these addresses and then waits for responding ACK segments. Because SYN/ACK segments are sent to nonexistent or unreachable IP addresses, they never elicit responses and eventually timed out (see Fig. 2).

※ Corresponding Author : Deokjai Choi, Address: (500-757) Advanced Network Lab.-445, Building 7 of Engineering, School of Electronic and Computer Engineering, Chonnam National University, Yongbong-dong, Buk-Gu, Gwangju, South Korea. 500-757, TEL : +82-62-530-3429, FAX : +82-62-530-3439, E-mail : dchoi@jnu.ac.kr
Receipt date : May. 27, 2014, Revision date : Aug. 1, 2014
Approval date : Aug. 21, 2014
[†] School of Electronics and Computer Engineering Chonnam National University (nugraha.muhammad@gmail.com)
[††] School of Electronics and Computer Engineering Chonnam National University (isyanaparamitha@gmail.com)
[†††] School of Electronics and Computer Engineering Chonnam National University (ardisragen@gmail.com)
[††††] School of Electronics and Computer Engineering Chonnam National University (dchoi@jnu.ac.kr)
[†††††] Korea Institute of Science and Technology Information (KISTI) (bscho@kisti.re.kr)
※ This research was supported by the Building and Operation KREONET and advancement of Service (K-14-L01-C03) of the Korea Institute of Science and Technology Information (KISTI) funded by the Ministry of Science, ICT & Future Planning.

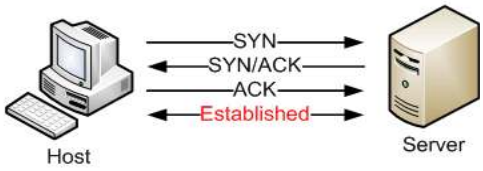

Fig. 1. TCP three-way handshake [3].

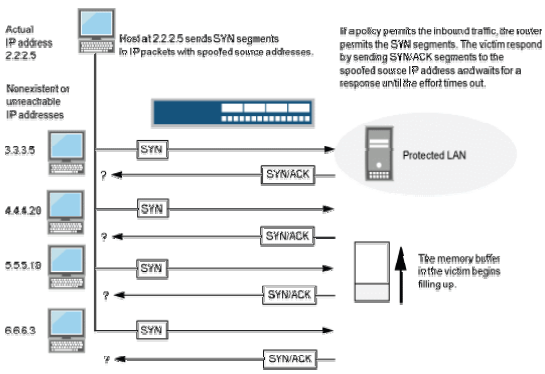

Fig. 2. TCP SYN flooding attack process [4].

Currently there are some methods to detect or mitigate SYN flooding attack such as SYN cache, SYN cookies, SYN proxying and SynDefender. All of these defense mechanisms are installed at firewall of victim server or inside victim server (see Fig. 3). Since defense line located at victim, network resources are also wasted because it transmitted a large number of packets, or packet flooding. Another method is using Flooding Detection System (FDS). FDS is, in some sense, a by-product of router infrastructure that differentiates TCP control packets from data packets.

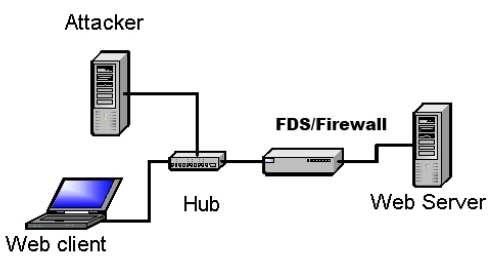

Fig. 3. SYN flooding defense mechanism in traditional network [5].

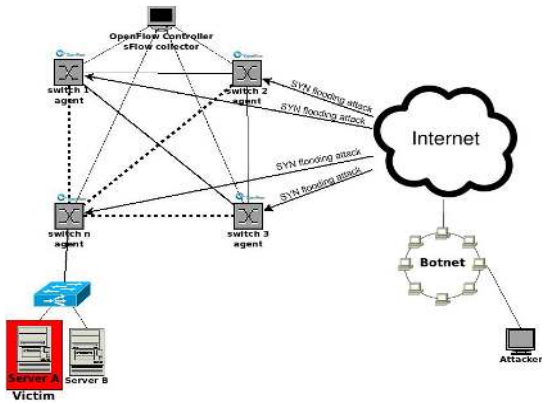

Fig. 4. Proposed model mechanism for SYN flooding defense using OpenFlow and sFlow.

FDS detects SYN flooding attacks at leaf routers that connect end hosts to Internet (see Fig. 3). Because of FDS located at leaf router, than router will consumes much resource.

Based on some cases in current method, this paper tries to overcome weakness by using Open Flow technology. By OpenFlow, attack packet can be detected early in some switches and resource consumption smaller than using traditional approaches. Fig. 4 shows that an attacker recruit many agents in Internet. agent of attacker will attack victim through each switch in our SDN network. In every switch, attack packet will be captured by sflow agent and sent to collector to be analyzed. For detection method system sum cumulative attack packet from every switch and collector will decide whether there is an SYN attack. If sum of packet is over certain threshold, controller will conclude there is an SYN attack.

Since Openflow network environment assumes controller has an pre-established connection between it and its agent, controller able to get number of SYN packet for specific destination node from each switch quickly. Because of that, controller also is able to calculate sum of those packets in early stage of attack.

The structure of this paper is as follows: in section 2 we will discuss about weak points of related works to be enhanced by our proposed mechanism.

Section 3 describes the design of our system include testbed and architecture, while in section 4 presents evaluation of our system to detect and mitigate DDoS attack by combining sFlow and OpenFlow. Finally, in section 5 concludes of paper and discusses future work.

2. RELATED WORK

Even though DDoS mechanism is widely understood by developers, its detection is very hard task because it is difficult to distinguish between normal traffic and useless packets, sent by compromised hosts to their victim. Rodrigo Braga et al. [6] presents a lightweight method for DDoS attack detection. He showed that his technique extracts feature of interest with a low overhead when compared to approaches based in the KDD_99 dataset. The shortage of his technique is it cannot apply statistical method to analyze port of OpenFlow switches to precisely determine which host launches attacks.

As explained by Junsang, Park et al.[7], in they paper they propose an SNMP-based lightweight and fast detection algorithm for traffic flooding attacks, which minimizes the processing and network overhead of the detection system, minimizes the detection time, and provides high detection rate. For detection algorithm consists of three consecutive stages. The first stage determines the detection timing using the update interval of SNMP MIB. The second stage analyzes attack symptoms based on correlations of MIB data. The third stage determines whether an attack occurs or not and figure out the attack type in case of attack. To use this method is not effective and efficient because his method need consume much resource.

The essential difference between normal stream and attack stream is that attack stream includes lots of spoofing IP address packets, which cannot build normal connection with destination host. Thus DDoS attack stream would include lots of one-way connection (OWC) [8]. One-way connection density (OWCD) not only identifies existence of DDoS but also indicates density of abnormal connection. OWCD series are a stationary series whose mean is not equal to 0. Calculating cumulative Euclidean distance between OWCD of present network flow and reference series, not only can be used to distinguish existence of DDoS, but it also detect attack intensity which is important to network administrator to detect and defend DDoS attack.

As Haining Wang et al.[9] did to detect SYN flooding attack, it puts mechanism in leaf router which connects host to Internet. Its detection mechanism can be applied only on leaf router, so it will consume much resource when SYN flooding attacks come and detection time is quite long.

To overcome those weakness we will propose mechanism to detect and mitigate SYN flooding attack by utilizing OpenFlow and sFlow. By utilizing those two features we can detect and mitigate SYN flooding attack earlier and detection mechanism will decrease resource consumption when attacks come.

3. PROPOSED SYSTEM

The objective of designing system to detect and mitigate SYN flood attack is to improve performance of detecting from only one router in leaf router to many routers. Thus, by using OpenFlow we can implement this mechanism.

3.1 SYN Flood Attack Detection Architecture

To design SYN flood attack detection architecture we have to consider resource consumption and detection time. In our system we use SDN technology by utilize OpenFlow and sFlow as tools. We can see in the Fig. 4, we use four OpenFlow switches that support sFlow agent that is controlled by one controller. So, by using four agents to detect SYN flood attack we can decrease re-

source consumption than using traditional technology which use just one machine.

3.2 sFlow Detection Mechanism

sFlow is a tool for traffic monitoring in SDN technology. By utilizeing sFlow, we can detect SYN flood attack by configuring and program it from controller. sFlow agent uses sampling technology to capture traffic statistics from device which is monitored, and sFlow agent also use sFlow datagram to immediately forward sampled traffic statistics to a sFlow collector for analysis.

sFlow agent, samples packets randomly with a 1-in-N probability - when packet is sampled, then sample will sent to collector [10]. Independently sFlow agent sends interface counters periodically - this is configured at regular intervals - by default every 30 seconds. Every second sFlow agent examines list of counter sources and sends any counter that need to be sent to meet sampling interval requirement.

For example we configure sFlow in OVS using commands shown in Table 1. In those syntax we configure agent whose name is br0 with IP 10.0.0.2 and collector IP is 192.168.2.10:6343. sFlow flow max-header is set as default 128 bytes. sFlow flow max-header used to set maximum number of byte of a packet. For sampling rate we set 500 because we use Link Speed 100Mb/s [11]. Number 500 of sampling rate means is every 500 packet which captured by agent will be sampled one packet to be sent to collector. For polling interval we set 10 second, so agent will send datagram to collector every 10 second.

Finally, we set threshold to detect attack as shown above.

Table 1. Configuration format of sFlow in OVS

```
Ovs-vsctl  --  -id=@sflow create sflow
    agent=${10.0.0.2}
target=\"${192.168.2.10}:${6343}\" header=${128}
    sampling=${500}
polling=${10}  --  set bridge br0 sflow=@sflow
```

3.3 Detection Algorithm

As we mention in introduction part that for detection methodology we sum cumulative attack packet from every switch and collector that can make decision to block packet when sum of cumulative attack packet over threshold (equation 1).

$$\Sigma F_s > T \tag{1}$$

$$R = \frac{\Sigma F_s}{\Sigma S_x} \tag{2}$$

Where ΣS_X is total number of switch, F_s is number of SYN flood packet which is found at switch s, T is threshold, R is average resource consumption of each switch. Resource consumption depends on the number of switches. For example we assume that if there are 14.000 SYN packets coming per second we indicate that traffic is SYN flood attack [12]. If we use one switch as traditional approaches, system will indicate that traffic is SYN flood attack when attack above 14.000 SYN per second. But with four switches like in Figure 4, system will detect SYN flood attack when sum of participating switches is crossing 14,000 which is much faster and consumes resources of each switch smaller. Graphic comparison shown in Fig. 5.

Last, we also should consider about redundant packet. As we describe before in our system we use four switches and each switch connect each

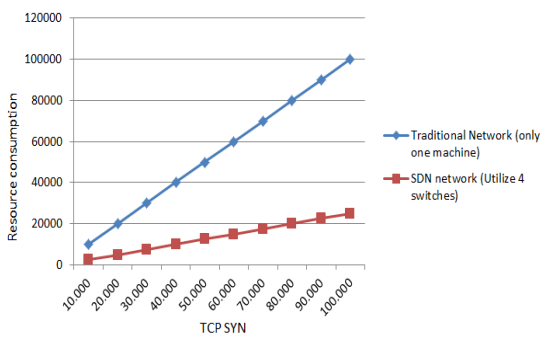

Fig. 5. Resource consumption comparation between traditional and SDN network.

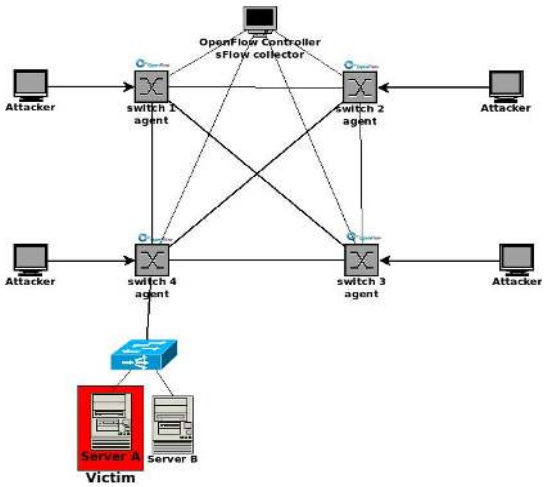

Fig. 6. Design architecture of SYN Flood mitigation.

other. To avoid packet redundancy we can configure each switch, so every packet which comes from our switch (switch1, switch2, switch3, switches n) does not counted by sFlow agent. For example there are n packets from internet come to switch2. To go to target n packet should through switch4. To be able to avoid packet redundancy at switch4, we have to configure switch4 so it does not count incoming packet from switch 1, 2, 3.

4. EVALUATION

4.1 Testbed

To implement our system, first we need to construct simple testbed for simulation how system work. Component what we need to construct testbed is mininet and miniedit as emulator, Open vSwitch as a virtual switch that support OpenFlow and sFlow. Since we want to implement SDN concept, so we use controller that can control each OpenFlow switch in mininet. For OpenFlow controller we use Floodlight controller which able to remote default controller in mininet and for sFlow collector we use sFlow-RT collector. Also, we use KVM virtual machine manager to run some virtualization component. The entire feature installed on Linux Ubuntu 12.04 LTS. Fig. 6 shown design architecture of our system to mitigate SYN flood attack.

4.2 Performance

To evaluate performance of our system, we design topology as shown at Fig. 6. We use four switches which build in mininet and one controller in different machine. Inside controller we put Floodlight as OpenFlow controller and sFlow-RT as sFlow collector. For sampling rate we set in 1:500, it is mean every 500 packets which through each switch it will take 1 packet to bundle in datagram to send to collector for analyze. After all of components ready as designed in Fig. 6, we generate SYN flood packet to attack victim, and Fig. 7 shows wireshark packet capture screen.

Flow sampling obtains information of flow specified service, whereas counter sampling obtain traffic statistics on an interface. So, by utilizing flow and counter sampling we can configure and calculate them to detect SYN flood attack by sampling mechanism as sFlow did.

We can see in Fig. 8 that traffic got from cumulative of each switch and we set threshold at 1M,

No.	Time	Source	Destination	Protocol	Info
1	0.000000	10.131.87.112	10.131.87.111	TCP	14550 > http [SYN] Seq=0 Win=512 Len=0
2	0.000002	10.131.87.112	10.131.87.111	TCP	14551 > http [SYN] Seq=0 Win=512 Len=0
3	0.000003	10.131.87.112	10.131.87.111	TCP	14552 > http [SYN] Seq=0 Win=512 Len=0
4	0.000004	10.131.87.112	10.131.87.111	TCP	14553 > http [SYN] Seq=0 Win=512 Len=0
5	0.001894	10.131.87.112	10.131.87.111	TCP	14554 > http [SYN] Seq=0 Win=512 Len=0
6	0.001896	10.131.87.112	10.131.87.111	TCP	14555 > http [SYN] Seq=0 Win=512 Len=0
7	0.003709	10.131.87.112	10.131.87.111	TCP	14556 > http [SYN] Seq=0 Win=512 Len=0
8	0.004251	10.131.87.112	10.131.87.111	TCP	14557 > http [SYN] Seq=0 Win=512 Len=0
9	0.007647	10.131.87.112	10.131.87.111	TCP	14558 > http [SYN] Seq=0 Win=512 Len=0
10	0.007648	10.131.87.112	10.131.87.111	TCP	14559 > http [SYN] Seq=0 Win=512 Len=0

Fig. 7. Capture traffic of SYN flood attack.

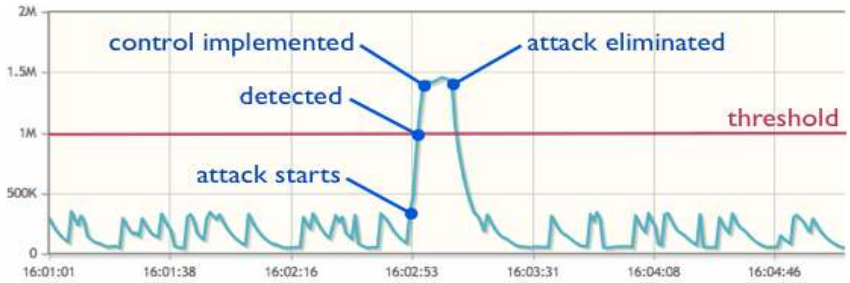

Fig. 8. Detection and mitigation attack process.

it means when cumulative number of packet from each switch is over 1M, collector will send alarm to OpenFlow controller, and OpenFlow agent will modify flow table to mitigate attack. It takes around 4 seconds to detect attack, and takes around 10 seconds to eliminate attack packet, and for all time process from attack starts until traffic return to normal it takes around 14 seconds. So, after all traffic eliminated, traffic state returns to normal state and mitigation process is done.

5. CONCLUSION

This work present SYN flood attack detection and mitigation by using OpenFlow and sFlow. We use four switches in our experiment and sum cumulative attack packet from every switch and controller will decide it whether those traffics indicate as attack or not. For result we showed that our mechanism using OpenFlow and sFlow is better than using traditional network because OpenFlow utilize many switch for defense which is controlled from controller. So, by utilizing OpenFlow and sFlow we can decrease resource consumption and increase time detection. For future work, we plan to find mechanism how sFlow and Openflow can detect and mitigate directly in real-time.

REFERENCE

[1] C.N. Maregeli, *A Study On TCP-SYN Attacks And Their Effects on A Network Infrastructure*, Master's Thesis of Delft University of Technology, 2010.

[2] Transmission Control Protocol, http://www.ietf.org/rfc/rfc793.txt. (Accessed May, 20, 2014)

[3] TCP Three-way Handshake, http://www.georgecoding.com/index.php/tcpdump-and-3-way-handshake/ (Accessed May, 20, 2014).

[4] TCP SYN Flooding Attack Process, http://www.juniper.net/techpubs/en_US/junos12.1/topics/concept/denial-of-service-network-syn-flood-attack-understanding.html (Accessed May, 20, 2014).

[5] SYN Flooding Defense Mechanism in Traditional Network, http://www.tech- mavens.com/synflood.htm (Accessed May, 20, 2014).

[6] R. Braga, M. Edjard, and A. Passito, "Lightweight DDoS Flooding Attack Detection Using NOX/ OpenFlow," *Proceeding of 35th Annual IEEE Conference on Local Computer Networks*, pp 408-415, 2010.

[7] Jun-Sang Park, Sung-Yun Kim, Dai-Hee Park, and Myung-Sup Kim, "Design and Implementation of an SNMP-based Traffic Flooding Attack Detection System", *The Korea Information Processing Society Transactions*, 1598-2858, prep, pp, 2009.

[8] Tu Xu, Da Ke He, and Yu Zheng, "Detectiong DDoS Attack based on One-Way Connection Density," *Proceeding of 10th IEEE Singapore International Conference on Communication Systems*, pp. 1-5, 2006.

[9] H. Wang, D. Zhang, and K.G. Shin, "Detecting SYN Flooding Attacks," *Proceeding of*

Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 1530-1539, 2002.

[10] sFlow Version 5, http://sflow.org/sflow_version_5.txt (Accessed May, 20, 2014).

[11] sFlow Sampling Rate, http://blog.sflow.com/2009/06/sampling-rates.html (Accessed May, 20, 2014).

[12] Hot Spares for DoS attacks, http://static.usenix.org/publications/login/2000-7/apropos.html (Accessed May. 20, 2014).

Muhammad Nugraha

He is master student in Computer Science at Chonnam National University. He received B.S. in Informatics Engineering from Ahmad Dahlan University, Indonesia. He is interest in Computer Networks.

Ardiansyah Ardi

He is an Internet of Everything Enthusiast from Indonesia with more than 5 years of lecture, research, and professional experience engaged in the development of electronics and computer technology in age of Internet of Everything. He received B.S. in Computer Engineering from University of Indonesia, Indonesia, and M.S. degree in Computer Science from Chonnam National University.

Isyana Paramitha

Curently she is undergraduate student in Computer Science at University of Indonesia. She is interest in Computer Networks.

Deokjai Choi

He is professor in Departement of Electronics and Computer Engineering, Chonnam National University, South Korea. He received B.S. degree in Department of Computer Science, Seoul National University, in 1982. He received M.S. degree in Department of Computer Science, KAIST, South Korea in 1984. He received PhD degree in Department of Computer Science and Telecommunications, University of Missouri, Kansas City, USA in 1995. He interest on research spans from context awareness, pervasive computing, sensor network, future Internet and IPv6.

Buseong Cho

He is senior researcher in Korea Institute of Science and Technology Information (KISTI). He received B.S. degree in Electrical and Computer Engineering from Sungkyungwan University, South Korea, M.S. degree in Computer Networks from Sungkyungwan University, South Korea.