논문 16-19-02-02

KNOM Review '16-02 Vol.19 No.02
https://doi.org/10.22670/knom.2016.19.2.13

레거시 미들박스에 의한 소프트웨어 정의 네트워크 보안 위협 분석

뉴엔 신 응억[◆], 뉴엔 반 퀴엣[*], 김 경 백[°]

Analysis of Legacy Middlebox Driven Security Threats of Software Defined Network

Sinh Ngoc Nguyen[◆], Van-Quyet Nguyen[◆], Kyungbaek Kim[°]

요 약

소프트웨어 정의 네트워크(SDN)는 쉽게 네트워크를 설정하고 저비용으로 네트워크를 관리하는 것과 같은 네트워크 관리를 위한 고급 기능을 제공한다. 그러나, 네트워크 플로우 관리를 위해 SDN을 이용하는 것은 새로운 보안 문제들을 발생시켰다. 이중 하나는 NAT, Proxy, IDS/IPS, LB, FW, IPSec Gateway와 같이 다양한 네트워킹 또는 보안 목적을 위해 사용되었던 기존 미들박스와 관련된 보안 문제이다. 기존 미들박스는 자체 정책에 따라 패킷 정보를 동적으로 변화시키기 때문에, SDN 컨트롤러는 해당 미들박스를 지나는 네트워크 플로우를 위한 구체적인 룰을 적용하기 힘들다. 이 논문에서는 기존 미들박스와 관련된 소프트웨어 정의 네트워크에서의 보안 문제들을 살펴보고, 가능한 대응방법들을 기술한다.

Key Words : Software Defined Network, Middlebox, Security

ABSTRACT

Software Defined Network (SDN) provides advanced methods for administrating network systems such as easy configuration of networks in remote and effective management of networks with low cost. However, using SDN to manage network flows has issued several new security problems. One of them is related to a legacy middlebox which is used for wide variety of networking and security purposes such as network address translation (NAT), proxy, network intrusion detection/prevention system (IDS/IPS), load balancer (LB), firewall (FW), and IPSec gateways for virtual private network (VPN). Because a legacy middlebox modifies the information of a packet dynamically based on their policies, it is hard that a SDN controller applies an accurate rule to a network flow which goes through the middlebox. In this paper, we expose several security problems of SDN related to a legacy middlebox, and describe possible approaches to resolve those problems.

※ 이 논문은 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(R0110-16-1001, IoT 기기 서비스 거부 공격 방어를 위한 리소스 보호용 시큐어 하드웨어 컨테이너 기술)

◆ Chonnam National University, Department of Computer Science, sinhngoc.nguyen@gmail.com
* Chonnam National University, Department of Computer Science, quyetict@gmail.com
° Corresponding Author : Chonnam National University, Department of Computer Science, kyungbaekkim@jnu.ac.kr

논문번호 : KNOM2016-02-03, Received September 23, 2016; Revised October 9, 2016; Accepted November 11, 2016

Ⅰ. Introduction

Computer networks are dynamic and complex and configuring and managing them reach to a big challenge. A network system includes a large number of routers, switches, and numerous types of middleboxes with many types of events occurring simultaneously. Network operators are responsible for configuring a network to enforce various high-level policies, and responding to the wide range of network events (e.g., traffic shifts, intrusions). A configuration of a network remains incredibly difficult because implementing these high-level policies require specifying them in terms of a distributed low-level configuration.

Nowadays, with the growing of internet, SDN is emerging as a new network technology to support demands of users. SDN separates between Control Plane and Data Plane that make the network more flexibility[1]. It controls a network system by setting up some rules into switching devices to direct flows of a network. The SDN controller has a central view of a network, it can make a decision to change the flows of a network when failures happens on the network, so it enhances controllability as well as reactivity of a network.

In legacy network systems, middleboxes are used for various networking and security purposes such as Network Address Translation (NAT), Proxies, Network Intrusion Detection/Prevention Systems (IDS/IPS), Load Balancing (LB), firewalls (FW), and IPSec gateways[2]. A middlebox monitors packets and sometimes it changes the information of packets depending on its own policies. The modification of packets complies with the functionality of a middlebox, but this characteristic issues problems when SDN is applied into legacy network with middleboxes. SDN requires exact information of packets when we want to apply a policy to network flows, but it is difficult to have the consistent view of the information of packets in SDN with legacy middleboxes.

In SDN, the control plane represents all of logics and switches, and it is responsible for making forwarding decision. The SDN controller which is the component of the control plane checks the current status of a network and generates new rules for each flow to be forwarded to which port of a switch. This checking procedure examines the information of a packet such as IP address, sources port, destination port on the header of packets. While conducting this checking procedure, information of packets should be passed correctly to a SDN controller[11][12]. However, several functionalities of middleboxes such as NAT or proxy server may modify some parts of packets without any notice to SDN controller. A middlebox often changes the header of a packet which holds the vital information for finding the source and destination of the packet, and the modification of a packet may be misunderstood by a SDN controller. Tne non-transparent manner of legacy middleboxes prevents detail explanation of why and what happened inside the middleboxes to a SDN controller, and the controller may not fully understand what happens in a network and can not generate proper rules for network flows. Additionally, this misunderstanding leads security issues on SDN as well such as bypassing firewalls and accessing network resources illegally.

In this paper, we consider the security problems driven by legacy middleboxes in SDN. we inspect the detail of processes and factors of the usage of legacy middleboxes in SDN. Also, we propose possible solutions such as how to maintain and improve the security in SDN with legacy middleboxes and how to interact between lagecy middleboxes and SDN controlelrs.

The rest of the paper is organized as follows. In section II, we describe the basic of legacy middleboxes and SDN, and explain security issues of SDN. We show several security problems of SDN related to legacy middleboxes in section III and propose possible solutions in section IV. In section V we provide some discussions related to the proposed solutions, we conclude in section VI.

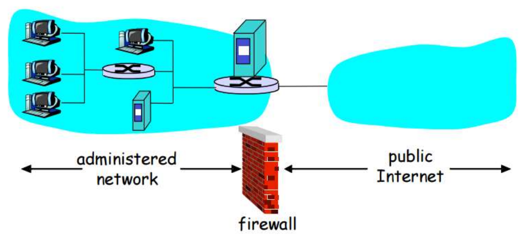

그림 1. 방화벽으로서의 미들박스
Fig. 1. Middlebox as a Firewall

Ⅱ. Background

1. Middlebox

A middlebox is a computer networking device which filters, transforms, inspects, or manipulates the packets of network traffic for various networking purposes. There are two famous examples which show the usage of middleboxes in networking; firewalls and network address translators (NAT).

The first example of a middlebox is a firewall. Figure 1 shows the mechanism of a firewall in a network. A firewall monitors and gives proper policies to each packet, and filters unwanted or malicious packets coming from outside network. With a firewall, an administrator of an internal network can isolate the internal network from internet by setting up the rules to block or allow the packets from outside network or vice versa.

The second example of a middlebox is NAT. As shown in Figure 2, a NAT modifies the information of packets in order to remap the internal IP address (private IP such as 10.0.0.1) into the external IP address (public IP address such as 138.76.29.7). The host address is unique only within the internal network, and a NAT converts unregistered internal address schemes into registered addresses before forwarding packets to public networks[13]. This technique is useful in case of rerouting traffic in IP address without readdressing every host.

Network routers often have integrated firewall, NAT or other functionalities, but sometimes dedicated middleboxes are preferred. According to this, how to use middleboxes still gets challenges and criticisms.

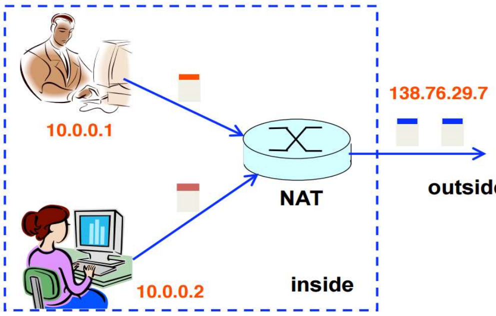

그림 2. 네트워크 주소 변환기로서의 미들박스
Fig. 2. Middlebox as a Network Address Translator (NAT)

2. Software Defined Network

SDN provides advanced methods for managing network systems. SDN simplifies operations and configurations of networks, and it also simplifies the functionality of network devices which do not need to understand the complex status of a network but simply accept instructions from a SDN controller. According to these characteristics, SDN has authentic features such as centralized intelligence, abstraction and programmability. A SDN controller has a global view of a network and it can make a decision to setup a flow based on the condition of the entire network. Also, SDN supports abstract feature, and it ensures the portability and future proofing in network services and network devices. The other feature of SDN is programmability. That is, SDN provides APIs for various application to interact with SDN, and network can achieve innovation and differentiation.

Conceptually, SDN has three separated layers for application, control, and infrastructure like Figure 3. Control layer stays at between application layer and infrastructure layer, and it can communicate to both layers through control plane interfaces (CPI). The control layer receives a policy from the application layer and sets up this policy into the infrastructure layer. Besides, the application layer provides APIs for users to interact with SDN easily.

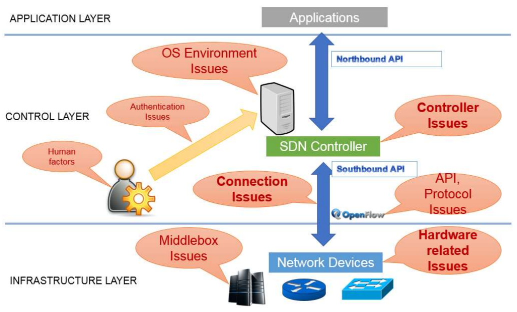

그림 3. SDN의 보안 이슈의 개요
Fig. 3. Overview of Security Issues of SDN

The infrastructure layer includes network elements such as the router, switch, hub, bridge, and so on. In a traditional network, those devices possess algorithms for forwarding packets and policy based decision-making. These algorithms depend on vendors, and they are static and restrict the flexibility and adjustability of network management. In contrast, SDN decouples network algorithms and networking equipments. The network algorithms are implemented in a SDN controller on the control layer with regular programming languages. Network elements receive control instructions from a SDN controller and simply abide the instructions to forward the packets.

The control layer is the core of SDN which provides the centralize view of a network. A centralized SDN controller on the control layer provides an overview of whole network and enhanced tracking of low-level of traffic. This allows that SDN controls data flows and dynamic forwarding decision based on the condition of the entire network, and SDN easily modifies forwarding schemes on the entire network at the same time by generating and transferring rules for handling packets. A SDN controller uses OpenFlow protocol to transfer rules from controller to network switches. All of complex algorithms related to forwarding, switching, and routing are programmed as an application on a SDN controller.

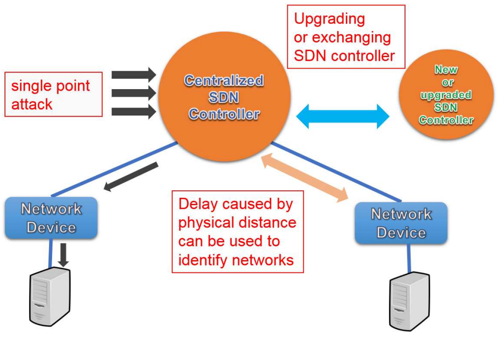
Upgrading or exchanging SDN controller
single point attack
Centralized SDN Controller
New or upgraded SDN Controller
Network Device
Delay caused by physical distance can be used to identify networks
Network Device

그림 4. SDN 컨트롤러 관련 보안 이슈
Fig. 4. Security Issues related to a SDN Controller

The application layer is the top layer of SDN, and it includes variety of network operation tools and user interfaces which support management and configuration of a network. It simplifies the complexity of the lower layer network, and it supports APIs for user to interact with SDN.

3. Security Issues of SDN

With the growing of internet, a computer network becomes more dynamic and more complex. A traditional network may not appropriate for management and configuration in a future network system. SDN can be a good replacement to improve the performance in operation and management of the future network.

Even though SDN provides promising features such as centralized control, flexibility, abstraction and programmability, multiple layers and open APIs of SDN lead new security issues. Figure 3 shows an overview of security issues of SDN such as controller issue, connection and protocol issue, hardware issue, operation system issue, and authentication issue.

Figure 4 indicates the security issues related to a SDN controller. A SDN controller is the core of management functionality of SDN, and it can be a perfect target for attackers to subvert a SDN system. That is, the SDN controller becomes a single point of failure. Sometimes, a SDN controller requires to be exchanged for recovery purposes or upgraded for better functionalities. However, during these actions, a SDN controller may lose the consistency of defending rules against attackers. Also, as the nature of SDN the distance from network devices to controller causes additional delays and attackers may recognize the existence of a SDN controller in a network. To resolve the security issues related to controllers, the concept of distributed SDN controllers should be adopted.

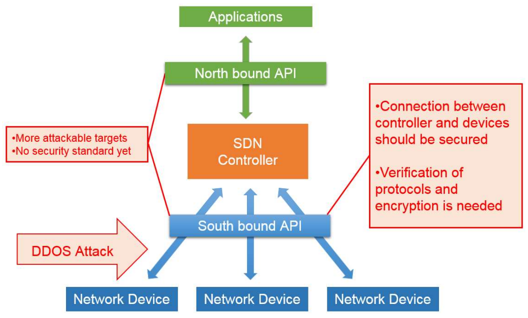
Applications
North bound API
SDN Controller
•More attackable targets
•No security standard yet
•Connection between controller and devices should be secured
•Verification of protocols and encryption is needed
South bound API
DDOS Attack
Network Device
Network Device
Network Device

그림 5. 커넥션 및 프로토콜 관련 보안 이슈
Fig. 5. Security Issues related to Connection and Protocol

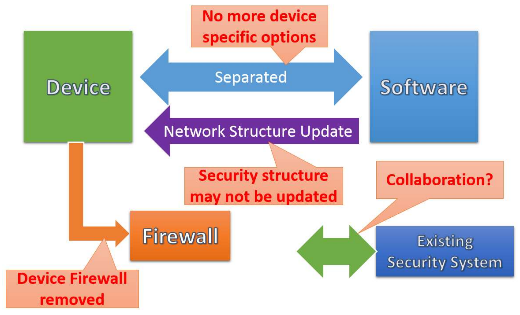

그림 6. 하드웨어 관련 보안 이슈
Fig. 6. Security Issues related to Hardwares

Connections and protocols are important manner for transferring data between network elements in SDN. Figure 5 depicts the security issues related to connections and protocols between controller to devices and controller to applications. There are more connections to manage a network, and the attack points becomes more as well. Each connection uses open APIs, OpenFlow protocol[4], and XMPP(Extensible Messaging and Presence Protocol) for data transferring, but secure connection is not much considered. Also, because of many connections, the possibility of DDoS attack increases[3]. To resolve these issues, we need to enhance the security of connections in SDN by using secure protocols such as SSL[5] and to apply a filtering system to distinguish the malicious traffic.

Figure 6 shows the security issues related to hardwares of network devices. SDN achieves flexibility and programmability by separating complex control logics from network elements, but the network elements also lose the basic ability against attacks. Firewall functionality is removed from network devices and security structure may not be easily updated. Also the collaboration between network devices and exisiting security systems are challenging as well.

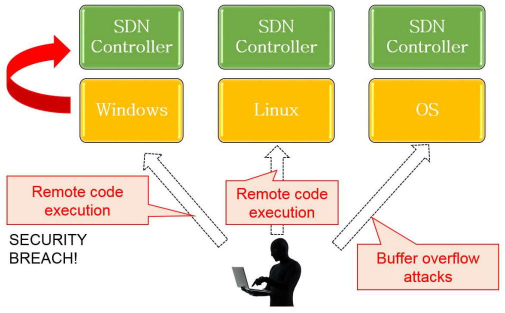

그림 7. 운영체제 관련 보안 이슈
Fig. 7. Security Issues related to Operating Systems

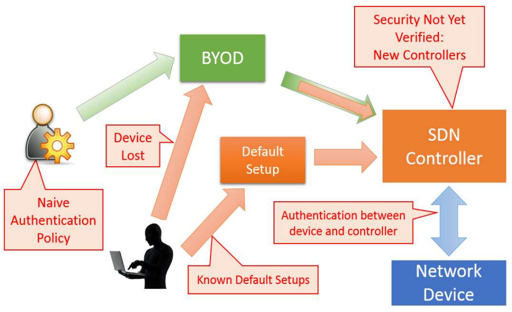
Security Not Yet Verified: New Controllers
BYOD
Device Lost
Default Setup
SDN Controller
Naive Authentication Policy
Authentication between device and controller
Known Default Setups
Network Device

그림 8. 인증 관련 보안 이슈
Fig. 8. Security Issues related to Authentication

Currently, a SDN controller runs on an operating system such as Linux and Windows like Figure 7. Then, the security breaches such as remote cote execution and buffer overflow can be exploited by attackers. To resolve this issue, we should have an appropriate strategy of operating systems for running application.

Figure 8 describes the security issues related to

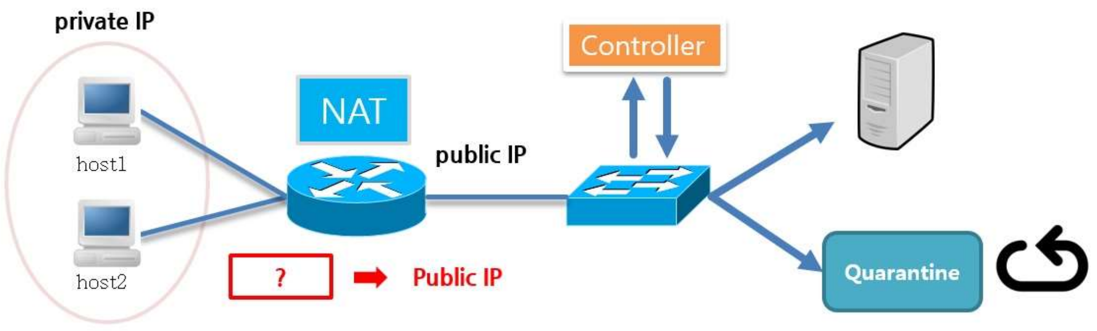

그림 9. NAT에 의해 이미 수정된 패킷의 원래 정보를 얻기 어려움에 따른 보안 이슈
Fig. 9. Security issues with difficulty of getting original information of a packet which has already modified by NAT

authentication. In SDN, there are many elements and they need to authenticate each other to ensure the secure connections. That is, we need authentication between a network device and a SDN controller as well as authentication between a SDN controller and another SDN controller. If the authentication process is too simple and uses a default setup, attackers may use BYOD (Bring Your own Devices) default setup to access a SDN controller. To resolve this issue, we need to make the authentication process for controllers and devices as complex as possible.

Besides these security issues, there is interesting security issues related to legacy middleboxes and they are explored in section III.

Ⅲ. Security Issues related to Legacy Middleboxes

To setup routing rules in a network, SDN requires correct information from a packet such as source IP and source port. In a normal case, it is easy to get this information from the header of a packet by forwarding it from a switch to a SDN controller. But in the case of a network which possesses legacy middleboxes, obtaining the correct information from a packet becomes hard, because a middlebox changes the header of a packet by its policies and it does not provide the reason why the packet has been changed. Also, it may not only disable load balancing, but also enable unintended or intended penetration of security policies generated by a SDN controller. In this section, we explore security issues in SDN network system related to middleboxes such as NAT and proxy.

1. Security Issues with NAT

Network Address Translation (NAT) is a technology which modifies some information in the header of a packet such as port number and source/destination IP address during transferring data through a router. NAT is used to remap a private network IP address of a host to public IP address. For example, we can reroute a network by using a public IP address without changing the host IP address.

In SDN, this modification of a packet by NAT middlebox makes a SDN controller hard to apply its policy to the packet correctly. In order to make a right decision, the SDN controller must have the original information of a packet. However, NAT modifies the private IP of a host to a public IP in the packet header, and the SDN controller can only obtain the modified information with the public IP. If a policy just checks information of the source IP and changes the flow of a packet to be quarantined or to be forwarded to a firewall, this policy may be failed to filter the target flow which has modified source IP address. This causes a security issue that a malicious traffic can pass firewalls. Figure 9 shows this security issue related to NAT middlebox. The mapping between a private IP

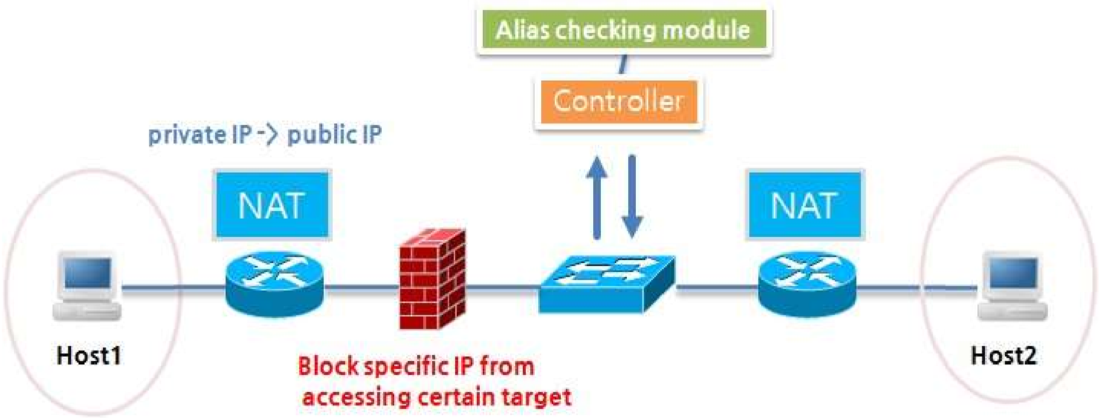

그림 10. NAT에 의한 룰 위반 체크의 복잡도 증가에 따른 보안 이슈
Fig. 10. Security issue with increase of complexity for checking rule conflicts by NAT

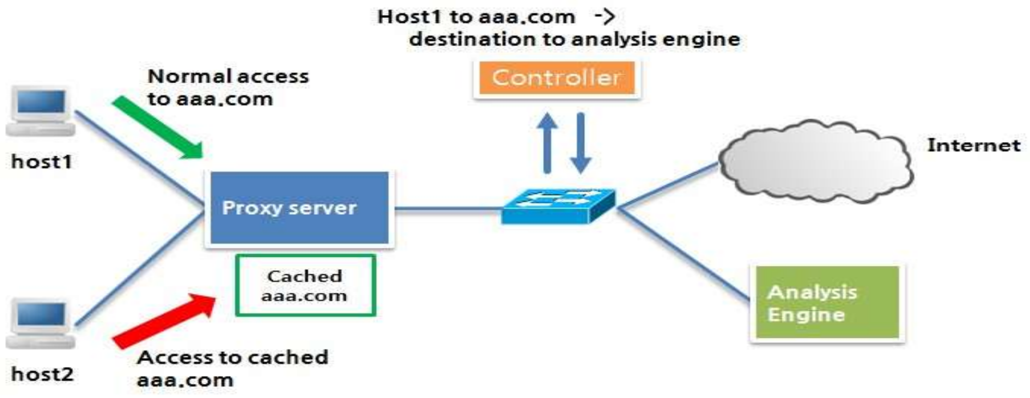

그림 11. 프락시에 의한 비정상적 데이터 접근에 따른 보안 이슈
Fig. 11. Security issue with abnormal data access caused by proxy

and a public IP is not informed to the SDN controller, the controller is hard to identify which one is coming from. So, the controller can not setup a right policy to the flow of a packet.

One more security issue with NAT middlebox is increasing the complexity of checking rule conflicts by SDN controller. Sometimes, controller adds a module which is used to check all flow rules and calculates whether there is any combination of rule conflicts or not[6]. As show in Figure 10, we may assume that there is a firewall which is located between controller and host1, and blocks a certain access. Even though the firewall works perfectly, the change of source and destination IP address of the rule in the controller can reach to security penetration. It means that a chain of rule can combine to enable a host to access the blocked target on firewall. The module of rule checking conflicts may detect the derivation of the rules for security penetrations. However, in the NAT setting, rule checking conflicts becomes more hard or complex because NAT modifies the information of packet automatically and dynamically. To check the rule conflicts correctly, it should check NAT IP matching rules, and it is possible only if NAT middlebox provides sufficient information.

Another thing which should be considered is monitoring the traffic in the network. Because, it is an essential part of making security which is related to decision making. For instance, there are many connections established to one host, we can assume that host is a suspicious target which should be analyzed. However, IDS cannot distinguish the original suspicious flow and the normal flow which one is modified by NAT, it hard to measure the exact number of connection. This wrong information of connection makes SDN controller hard to apply the policies for security.

2. Security Issues with Proxy

Proxy allows that clients in an isolated network indirectly connect to other network services. It is located between clients and a server. The proxy services receives a request from a client, gets the content from a server and store it into its internal cache. After this procedure, other clients can access to data from this cache without connecting to the server directly. And this help of proxy service provides more efficiency to users.

However, in the security purpose, this may cause some problems. In Figure 11, the controller wants to apply a rule for incoming packets such that the packets are forwarded to the analysis engine to detect and block packets from blacklisted host. If a network has a proxy middlebox between hosts and the analysis engine, some hosts may obtain some contents from a server without passing through the analysis engine. This case shows the possibility that an attacker can bypass the security system using proxy.

Also, another security issue of using a proxy middlebox is to hide the real identity of a user including the IP address of hosts. One of the reasons to use a proxy is that malicious packets cannot directly attack to host. However, it also hides the original packet and lacks the information to make a right decision for the whole network by a SDN controller

3. Security issues with closed manner of Middleboxes

A SDN controller is a crucial element in SDN, and it generates rules to change the flow of network by checking information which sent to centralized controller. This information includes some important parameters which are used to decide which policy should be applied. For some case of security, it requires historical information of flows. However, a middlebox cannot provide enough information to make such decisions. The controller should know the logs to explain why the packet has been changed and which source has been changed to generate the right decision for managing the whole network. To resolve these problems, we need to have useful information which is enough to create the right decision for controller.

Ⅳ. Possible Approaches

1. Approach of adding Information on Packet

To solve the security issues related to middleboxes, a SDN controller need to obtain the original information of packets. To get original information of packets, we can add more information to the packet modified by a middlebox. A middlebox becomes responsible for adding the useful information which is used by a SDN controller. A SDN controller operates the actions of a middlebox related to information processing and translate the information to apply right policies. To achieve this solution, a little changes in a middlebox, a controller and switches are required.

Figure 12 shows the appraoche to resolve the miss match information between a middlebox (NAT) and a SDN controller by sending the original information of a packet from a middlebox. We need a Generation Module on a middlebox (NAT) to add the mapping information of a packet. And we need a Reading Module on a SDN controller to understand the mapping information sent from the middlebox and let the controller recognizes the original source of the packet. That is, the original information of a packet is used to generate a right decision on the SDN controller. Also, we need to modify a little bit on a switch to capture modified packets which

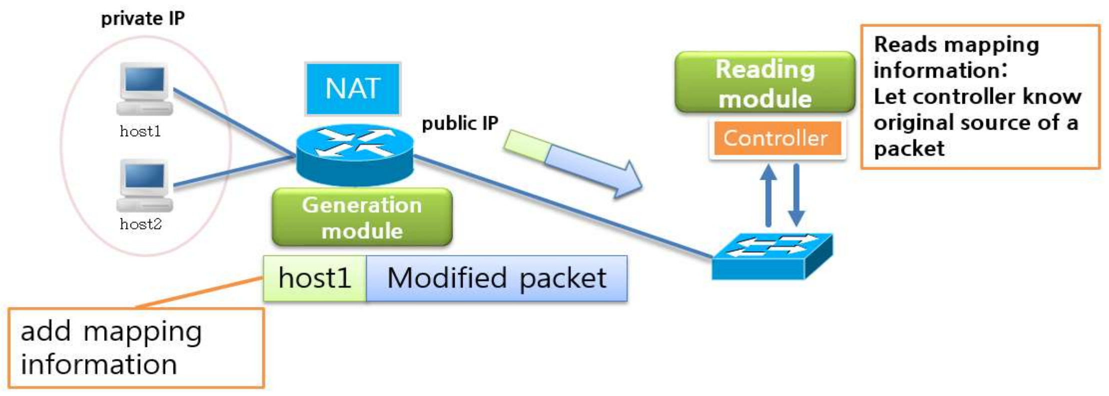

그림 12. 미들박스(NAT)에서 패킷의 원래 정보를 전달하는 방법
Fig. 12. Approach of sending the original information of a packet from a middlebox (NAT)

sent from a middlebox.

One solution to modify the packet on middlebox is FlowTags[7], which provides more useful information to the packet which controller desires. It looks like that we are observing something on the market, it will suggest tags, which includes the information needed for controller to generate the right decisions. A FlowTags is generated by a middlebox, and it contains information which will be used by the controller and other middleboxes to setup the policies. A FlowTags can be implemented in 64 bits because of the limitation in packet headers. The middleboxes follows the instruction from the controller when processing tags, and if there are no matching tags rule of the packet, the middleboxes can request controller through the southbound APIs and gets back the rule from controller.

In this approach, a controller provides tagging instruction rule for middleboxes, and tag translation for decoding information. In the switches, to capture the packet sending from a middlebox, we need to modify the rule on the OpenFlow to match the packet header. FlowTags suggest the structure which helps the switches and middleboxes act independently. Also, the compatibility with the existing machines have been considered for easier way to implement module to legacy middleboxes.

However, this system indicates some questions. Even though the tags are added to the packet, the controller may not understand this process. This misunderstanding cause blocking the functionality of the middleboxes. Also, even though it tries to modify the existing network, it requires additional modules to support tags. For reality, all of connected network should be modified to support FlowTags. And another thing is the delay which causes by tags processing. If we do not install the FlowTags rule to middleboxes in this structure, the overhead problem will come from installing instruction to middleboxes, it can add more burden to the performance.

2. Approach of centralized control of middleboxes

Another approach to resolve above problem is centralized control of middleboxes which have the centralized view and control the network by middleboxes. In this approach, we unify the instruction of the controller and middleboxes, and enables direct communication between them. This allows the controller has the overviews of whole network, and pushing the policies on the middleboxes directly. It includes the policy of middleboxes to change the destination packet, source packet, and processed packet.

There has been some researchers that focus on improving the performance of middleboxes using SDN[8][9]. However, centralized control plane of middlebox is not only used to improve the

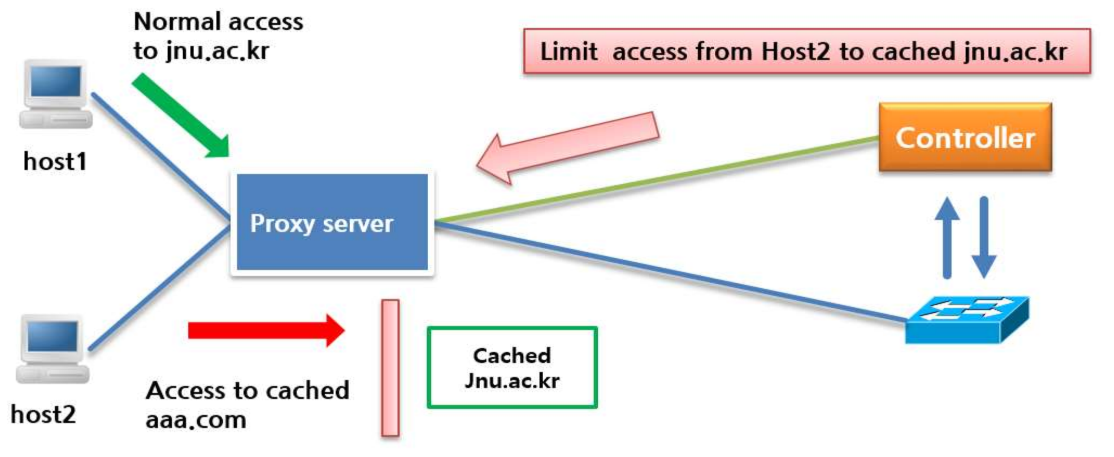

그림 13. 미들박스들의 중앙 제어 방법
Fig. 13. Approaches of centralized control of Middleboxes

performance and fault tolerance[10], but also to improve the decision making of the controller. It has full understanding about processing of middleboxes, so it can give the appropriate policies to middleboxes and setup the rule to switches. If there is a communication between middleboxes and controller, it can modify the rule automatically to support proper functions on the middleboxes.

Figure 13 depicts the centralized control of middleboxes by pushing the control rule from SDN controller to middleboxes. Middleboxes exactly know information of hosts, as well as the policy sent from the controller, so it can apply the combined rule to the flow of packet for filtering, forwarding, and so on. For instance, host1 allows to access jnu.ac.kr server, and host2 is not.

Although centralized control has many features, it gets some challenges. There are many kinds of middleboxes from other venders, so it is hard to provide a good interface between the controller and middleboxes. The structure and control logic are different from product to product and from venders to vendors. These heterogeneity makes it hard to provide a unified control, and may not fully support functions of middleboxes. Because, vendors do not want to open their structure and protocol to public, the controller may cause mis-communication to middleboxes.

The centralize control may be suffered by its structure when having a single point failure. In case of controller failure, every controlling including the middleboxes will be failed and malfunction happen.

In addition, working with centralized view, it can bring to new security threats. Because of the communication between a SDN controller and middleboxes, a malicious threat can access to the controller through the middleboxes. This prefers a proper protection for accessing the controller, and some information related to middleboxes and controller should be hidden. Also, performance issues can bring up by controlling with more dimensions. Adding more information and targets to the network can make the system control much more complex.

3. Hybrid Approach

From the above approaches, we can consider a hybrid approach. In this approach, the controller has a centralized view of whole network, and it can send the right decision to a middlebox through Middlebox APIs. In middleboxes, it has the exact information of hosts, and it is responsible for modifying the packet follow the policy sent from controller. This approach also resolves the security problem related to the communication between controller and

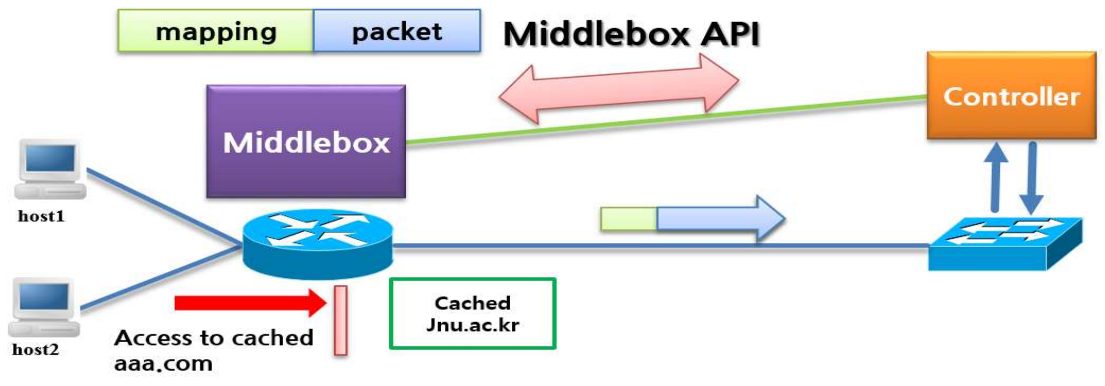

그림 14. 미들박스 관련 보안 이슈를 해결하기 위한 하이브리드 방법
Fig 14. Hybrid Approach for resolving security issues related to middleboxes

middleboxes. Middlebox APIs provides a safe communication between them. But, in order to have Middlebox APIs, the vendors should follow the standards from controller to unify access protocol.

In the Figure 14, a middlebox generates the new packets including mapping information to let controller know exactly about the isolated network. It also gets the policy from the controller through Middlebox APIs and setup the right decision for filtering or drop the packet at middlebox. From that, it can block the request from host2 to the jnu.ac.kr server.

V. Discussion

Malicious intrusion and modification at middleboxes under centralized environment can direct attackers to learn the information of the controller by analyzing the communication between the controller and middleboxes. Tagging the packet by adding information related to the mapping of middlebox reaches to the security problems. Capturing and analyzing the packet in the network helps attackers identify what kind of policies are operated by the controller, and this information can be used to bypass the policies.

There are several venders provide middleboxes with other functionalities. Using controller to control middleboxes without full understanding about internal security logic, it can cause security problems from causing malfunction of the middleboxes. Most of them use special way to detect and provide security, and this can be compromised by inappropriate control in the controller. Also, preventing middleboxes from collision among them should be considered.

In addition, the connection between controller and middleboxes should be encrypted. Because the attackers can capture the packet to get information to attack the middleboxes' logic and controller's policies. There is no verification for secure connection, and there is no standard that has been suggested for transferring the packet in the network.

The main purpose of middleboxes is to hide the information of traffic source to keep the traffic generator safe. If a middlebox has the central control and it can be observed by control application, this may violate the goal of the middlebox functionalities and it may bring to collision. In order to make the right decision for policy, some useful information is needed, but gaining process information should not harm the functionalities of middleboxes. This should be considered when implementing centralized system.

Ⅵ. Conclusion

SDN is a future network that brings us many benefits to overcome the lack of legacy network. However, applying SDN technique to a network with middleboxes gets some new security threats. In this paper, we explored security issues of SDN related to middleboxes which has non-transparent features, and describe some approaches which help us having a transparent view of middleboxes. The security issues of SDN related to middleboxes may be resolved by adding vital information of the packets during passing a middlebox, or centrally and directly controlling middleboxes by an SDN controller. In some cases, these solutions may harm the original purposes of middleboxes and cause to another additional security problems. So we need to carefully design the middleboxes which are well collaborated with SDN controllers.

References

[1] S. Scott-Hayward, G. O'Callaghan and S. Sezer, "Sdn Security: A Survey," 2013 IEEE SDN for Future Networks and Services (SDN4FNS), Trento, 2013, pp. 1-7.

[2] Tian, Chen, et al. "OpenFunction: An extensible data plane abstraction protocol for platform-independent software-defined middleboxes." Network Protocols (ICNP), 2016 IEEE 24th International Conference on. IEEE, 2016.

[3] Sharma, Shruti, et al. "Distributed Denial of Service Attack."

[4] Shalimov, Alexander, et al. "Advanced study of SDN/OpenFlow controllers." Proceedings of the 9th central & eastern european software engineering conference in russia. ACM, 2013.

[5] Hickman, Kipp, and Taher Elgamal. "The SSL protocol." Netscape communications corp 501 (1995).

[6] P. Porras, S. Shin, V, Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for OpenFlow networks," HotSDN, 2012, pp. 121-126.

[7] S. K. Fayazbakhsh, V. Sekar, M. Yu, and J. C. Mogul, "FlowTags: enforcing network-wide policies in the presence of dynamic middlebox actions," HotSDN, 2013, pp. 19-24.

[8] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella, "Toward software-defined middlebox networking," HotNets, 2012, pp.7-12.

[9] Z. A. Qazi, C. Tu, L. Chiang, R. Miao, V. Sekar, M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," SIGCOMM, 2013, pp.27-38.

[10] A. Gember, T. Benson, and A. Akella, "Challenges in unifying control of middlebox traversals and functionality," LADIS, 2012.

[11] H. Zhang and J. Yan, "Performance of SDN Routing in Comparison with Legacy Routing Protocols," 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Xi'an, 2015, pp. 491-494.

[12] T. Sasaki, C. Pappas, T. Lee, T. Hoefler and A. Perrig, "SDNsec: Forwarding Accountability for the SDN Data Plane," 2016 25th International Conference on Computer Communication and Networks (ICCCN), Waikoloa, HI, 2016, pp. 1-10.

[13] Shiuh-Pyng Shieh, Fu-Shen Ho, Yu-Lun Huang and Jia-Ning Luo, "Network address translators: effects on security protocols and applications in the TCP/IP stack," in IEEE Internet Computing, vol. 4, no. 6, pp. 42-49, Nov/Dec 2000.

뉘엔 신 응억 (Sinh Ngoc Nguyen)

2009: VietNam National University Ho Chi Minh City - University of Information Technology (B.S. Degree)

2016 ~ present : Chonnam National University, South Korea (M.S. Degree).

2013~2015: Software Engineer at Integrated Circuit Design Research and Education Center

2016~now : School of Electronics and Computer Engineering

뉴엔 반 퀴엣 (Van-Quyet Nguyen)

2005: Hung Yen University of Technology and Education. (B.S. Degree).
2011: Ha Noi University of Science and Technology (M.S. Degree).
2015 ~ present : Chonnam National University. South Korea (Ph.D Degree).
2009~2015: Lecturer in Hung Yen University of Technology and Education.
2015~now : School of Electronics and Computer Engineering.

김 경 백 (Kyungbaek Kim)

1999년 한국과학기술원 전기 및 전자공학과 학사 졸업
2001년 한국과학기술원 전기 및 전자공학과 석사 졸업
2007년 한국과학기술원 전기 및 전자공학과 박사 졸업
2007~2011년 University of California Irvine, 박사후 연구원
2012~2015년 전남대학교 전자컴퓨터공학부 조교수
2012년~현재 전남대학교 전자컴퓨터공학부 조교수
<주관심분야 : 분산시스템, 미들웨어, 피어투피어/오버레이 네트워크, 소셜 네트워크, SDN>