Middlebox Driven Security Threats in Software Defined Network

Sungmin Hwang and Kyungbaek Kim

Dept. Electronics and Computer Engineering
Chonnam National University
Gwangju, Republic of Korea
Sungmin1511@gmail.com, kyungbaekkim@jnu.ac.kr

Abstract

As future internet, Software Defined Network (SDN) has brought special benefits which are hard for legacy network system by separating control plane from data plane. A SDN controller manages network flows in centralized manner by applying proper rules dynamically on each flow. However, this distinct feature of SDN has brought several security issues. One of these issues is related to middleboxes which are commonly used to secure or manage the networks in legacy network systems. Middleboxes modify the information of packet dynamically based on their own policies, but the modification is hidden from outside network components because of the closed structure of middleboxes. According to this, a SDN controller does not recognize the modification of packets and may apply wrong rules to a flow. In this paper, we explore possible security problems caused by middleboxes and possible solutions for the problems.

Keywords-Software Defined Network; middlebox; security

I. Introduction

Due to various internet services and growth in internet user population, Software Defined Network (SDN) has been suggested to satisfy the demands which are hard for legacy networks to provide [1]. SDN separates control plane and data plane, and enables a SDN controller to control the network in centralized way. The SDN controller checks the network state and enforces flows by the policies it holds. This dynamic enforcement of flows helps to use the network more efficiently, and provides special benefits [1].

In legacy network systems, middleboxes are commonly used to provide distinct services such as filtering malicious traffics, network address translation, load balancing, and intrusion detection [7][8]. They examine the packet and sometimes, they change or filter the packet. This modification of packet has brought challenges to SDN which requires the exact information of packets to make a decision for applying which policies to which flows. Legacy networks also had this problem with middleboxes, but this became more serious in SDN because the main identity of SDN is to enforce policies for network management tasks [1].

In SDN, a switch pushes a packet to a SDN controller when it does not have any matched entry in its flow table. The controller decides where this flow will be headed to, or how the flow should be modified by checking the policies controller has. While this step is taking place, the value compared to match flow and policy is often stored in packet header. IP and ports can be examples. To apply the right policy and setup the right path, the information should be imported correctly to the controller.

However, middleboxes, such as Network Address Translator (NAT) and proxy server, can sometimes interrupt a SDN controller from having the exact view of the network. They often change the packet header that holds vital information for finding source of the packet, and this may harm right decision for applying policies. Whenever modification and filtering happens in middleboxes, non-transparent design of them prevents explanation why and what happened inside the middleboxes and this brings information block to the controller. This refers that the controller does not fully know what happens in the network and cannot make decision whether if the flow is appropriate to its security policies.

This characteristic of middleboxes can cause security breaches in some cases, and this shows need of inspecting detailed process and factors of these middlebox-driven security problems. Also, how to maintain or improve security under the circumstances and how to interact with middleboxes should be concerned. We explain what causes the problems in section2, and describe possible solutions in section 3, then discuss about more possible cases from middleboxes.

II. Middlebox-Driven Security Problems

A. Failures in Applying Correct Rules

SDN requires correct and stable information of a packet such as source IP to decide the path for the packet dynamically. In normal cases, it is easy to check the information by examining packet header information that an OpenFlow switch sends to a SDN controller. But in some cases where middleboxes are involved, this process becomes hard, due to difficulty of finding the real information. Since middleboxes change the packet header by their own decision and they do not provide why and from what the packet has been changed, this can cause not only disable load balancing, but also enable unintended or intended penetration of security policies. Here, we list possible middlebox-driven security threats.

Case of using NAT:

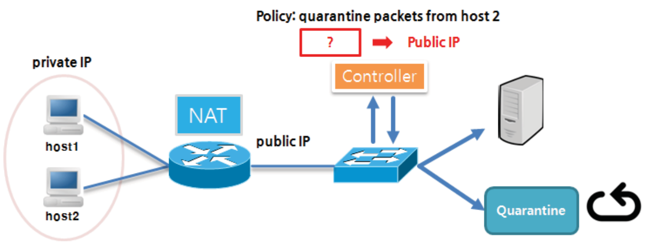

Fig. 1 Difficulty of getting right information from the packet which has already processed by NAT

NAT is a technology that rewrites TCP/UDP port number and source/destination IP of a packet during communication through the router. NAT makes modification to the packet, and this needs recalculation of IP and checksums. NAT is usually used for matching private network hosts to public IP, and this is very useful and needed for recent network environment. To SDN, this modification of a packet by NAT makes a controller hard to apply its policies to the packet. To make right decision, the controller needs to check the real information, but NAT translates IPs of hosts to public IPs by modifying the packet header. The controller can only identify the public IPs so that if the policy is to check the source and change the flow to make quarantine or send to firewalls, this checking process have a chance to fail the filtering the packet and installing the right rules. This can cause serious security problem since this can allow the access of hostile hosts. Fig. 1 shows how this can happen in network. Private IP and public IP matching is not informed to the controller, making controller hard to identify where the packet is coming from. This makes controller unable to apply the policy to the packet from host2.

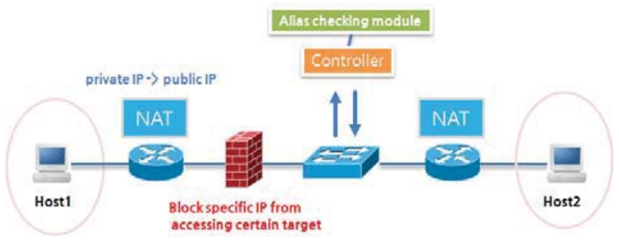

Fig. 2 NAT makes checking rule conflicts more complex

Another problem with NAT is increasing the complexity of checking rule conflicts by a SDN controller. Sometimes a SDN controller adds a module which checks every flow rules and calculates whether if there is any combinations of rules that derive rule conflicts [5]. For example, as Fig. 2 illustrates, it is assumed that a firewall is placed between the controller and hosts, and blocks a certain access. Even though the firewall works perfectly, the change of source/destination IP address by controller's rules can sometimes lead to security penetration. That is, chaining of the rules can derive a rule that enables a host to access the blocked target. The module of checking rule conflicts may detect the derivation of rules for security penetrations. However, in the setting with NAT, checking rule conflicts becomes more complex or impossible because NAT also dynamically modifies the information of packets. To check rule conflicts correctly the module should check NAT IP matching rules, but as discussed earlier, this is not possible if middle boxes do not provide sufficient information.

One more thing to consider is monitoring the traffic, since it is essential part for making security related decisions. For an example, if there are too many connections established by one host, we can assume the host as suspicious target which should be analyzed. But if IDS (Intrusion Detection System) cannot correctly distinguish the origin of a suspicious flow which is modified by NAT, it is hard to measure the exact count of connections. This incorrect information of number of connections makes a SDN controller difficult to apply its policies for security.

Case of using Proxy:

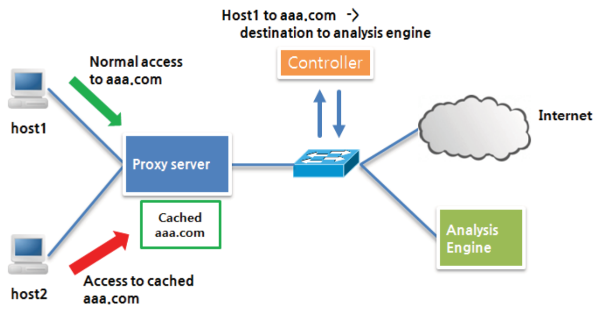

Fig. 3 Proxy server's cached data allows abnormal access to blocked path

Proxy enables clients indirectly connect to other network services. It is located between server and clients and stores the requested contents in the cache. From this cache, clients can access to stored data without connecting to the server again, and this helps proxy to provide more efficiency to users. However, for security purposes, this can cause some security problems. In Fig.3, the controller wants to apply rules for incoming packets that make the path which goes through a firewall to detect and block packets from blacklisted hosts. If the proxy lies between the firewall and host, blocked host need not to go through the detection system to access the server beyond the firewall. Instead, the host can just connect to the cached data, which has been originally requested by other hosts. This case shows that a proxy can provide attackers a way to bypass security systems.

Also, another common purpose of using proxy server is to hide the real identity of the user including the IP address of the host. One of the reasons of using this function is to provide the security to the hosts using proxy server. But in this case, the origin of the packet is hidden and this lacks the information for the controller to make right decision for whole network.

B. Closed structure

SDN's crucial feature is to dynamically change a path of a flow by checking information sent to a centralized controller. This information includes vital parameters that are needed to decide which policy the controller should apply. For some security related modules, it requires the history of a flow. The problem is that middleboxes which are widely used in networks do not provide enough information to make decisions. The controller should know the logs, why the packet has been changed and from what it has been changed, to make right decision for managing the whole network correctly. This non-transparent design is responsible for most of the middlebox-driven security problems. To overcome these problems, we need a way to have right information which will bring the right decision to the controller.

III. Possible solutions

A. Adding Information on Packet

To provide right information, one way is to add information to a packet and use it for applying rules. This direction enables least changes in switches and avoids direct interfaces between switches and middleboxes. In this case, middleboxes are responsible for adding and modifying the additional information. A SDN controller controls actions of the middleboxes that are related to processing the information, and translates the information to apply the policies. This structure requires modification of middelboxes and controllers, perhaps for small parts in switches to deal with information.

One example for this solution is FlowTags [4], which provide more information to the packet to support the vital information the controller needs. Like the things we can commonly observe in the market, it suggests tags, which reveal the information needed for the controller to make right decisions. A FlowTag is generated by a middlebox, and contains information which will be used by the controller and other middleboxes to apply policies. A FlowTag can be implemented in 6bit ToS/DSCP field due to space limitation in packet headers [4].

Middleboxes follow the action instruction rules from the controller when processing tags, and if there is no matching tag rule for the packet, it requests controller through the southbound API and gets the rule from the controller. Processing tags is done by the module which uses 'session-oriented', popular architecture among middleboxes, to support the actions and to maintain compatibility.

Controller supports tagging instruction rules for middleboxes, and tag translation for decoding information. For the switches, they just need to match on packet header fields defined by OpenFlow.

FlowTags suggest the structure which helps switches and middleboxes to innovate independently. Also, compatibility with existing machines has been concerned from using easier ways to implement modules to legacy middleboxes.

However, this system brings up some questions. Even though the tags are added to the packet and reports middleboxes' information, controller many not fully understand why they have done the process. This misunderstanding of the actions can block the functionality of the middleboxes by tagging only partial information.

Also, even though it tries least modification of existing network, it requires adding modules to support tags. For real use, whole connected network should be modified to support FlowTags.

One more thing to be concerned is delay caused by tagging process. With no pre-installed FlowTag rules to middleboxes in this structure, overhead, which comes from installing instructions to middleboxes, can add another burden to the performance.

B. Centralized control of middleboxes

Another approach is to have a centralized view and control of middleboxes in the network. Unified interface lies between controller and middleboxes, and enables direct communication between them. This allows the controller to have the whole view of the network, and make decision by examine what is taking place in the middleboxes. This includes middleboxes' policy to change the packet, source of packet, and processed packet. Some researches focus on improving the performance of middleboxes using SDN [2] [3]. However, centralized control plane of middleboxes can be used not only to improve the performance and fault tolerance [6], but also to improve the decision making of the SDN controller. Controller can understand the full process of middleboxes and can apply right policy and install rules to switches. If there is interactive connection between controller and middleboxes, the controller can also dynamically modify the rule to support proper functions to middleboxes.

Centralized control of middleboxes shows many features, but also leads to some challenges. There exist many kinds of middleboxes, which makes it hard to provide right interface between controller and middleboxes. Structure and control logic differs from product to product and venders from vendors. These heterogeneous results in very complex process to produce unified control, and may not fully support middlebox's functions. Also, vendors do not want to expose their internal structure and logic since it is their intellectual property, and this may cause miscommunication between controller and middleboxes. Centralized control may also suffer from its structure, when single point of failure happens. When the controller fails, every control including the middleboxe will be failed and malfunction.

However this can bring new security threats, since it works in centralized way. An access to controller can have the control of full connected middleboxes to modify the packets. This refers a need of proper protection for accessing the controller. Also, some middleboxes' main purpose is to hide the information for users, and this centralized control may enforce them to give up the benefits.

Also, performance issues can be brought up by controlling more dimensions. Adding more information and targets to control makes the control process much more complex.

IV. Discussion

A. Possible security threats of suggested solutions

Hostile intrusion and modification to middleboxes under centralized environment can enable attackers to learn the information of the controller by checking the communication. Observing and making queries to controller can help attackers to identify what kind of policy does the controller have, and this information can be used to ignore or bypass the policy. This also can be used for packet tagging cases.

Attempt to control without full understanding of internal security logic of middleboxes, can cause serious security breaches from causing malfunction of the middleboxes. Some of them use abnormal and special way to detect and provide security and this can be compromised by inappropriate control by the controller. Also, preventing middleboxes from collision between themselves should be concerned to secure their functionality.

In addition, Proper encryption of connection between controller and middle boxes should be supported, since the exposure of added information can be used to attack the middleboxes' logic and controller's policies. No verification of secure connection have been done yet, and standard has not been suggested either.

B. Keeping Functionality of Middleboxes

In some middleboxes, their main purpose is to hide information of the traffic source to keep the traffic generator safe. If the middleboxes can be centrally controlled and observed by control application, this may violate the goal of the middlebox functions and bring collision. To make the right decision for policy, some vital information is needed, but gaining information process should not harm the functionality of middleboxes. This should be concerned when implementing cetralized system.

V. Conclusion

SDN is suggested as a future internet and can bring us many benefits to overcome current network environment. However, the current middleboxes cause some security threats by providing insufficient information to SDN controllers. Listed problems are originated from non-transparent design, and solutions for them are to have a transparent view of middleboxes. This can be done by adding vital information to packets, or centrally and directly controlling middleboxes by a controller. In some cases, these solutions may harm purposes of middleboxes and can sometimes cause additional problems. This refers proper level of encapsulation is needed for the middle boxes while maintaining the functionality of SDN controllers. Examination of the suggested systems for a secure SDN network should be done to suggest more advanced system.

VI. Future Works

For next research, we will focus on designing the centralized control structure which centrally gets the information from every connected middleboxes. This will bring the listed challenges in section3, and to solve and reduce the problem, a new way to secure the information should be suggested. Also, we will try to find solution, which allows controller to get information while functionality and purpose of middleboxes remain.

Acknowledgements

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.

References

[1] Opennetworking.org, "Software defined networking: the new norm for networks," 2012, [accessed August 2014]

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

[3] 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.

[4] 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.

[5] 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.

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

[7] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," SIGCOMM, 2012.

[8] V. Sekar, S. Ratnasamy, M. K. Reiter, N. Egi, and G. Shi, "The middlebox manifesto: enabling innovation in middlebox deployment," HotNets, 2011.