Assessing the Impact of Resource Attack in Software Defined Network

Hiep T. Nguyen Tri, Kyungbaek Kim
Department of Electronics and Computer Engineering
Chonnam National University
Republic of Korea
Email: tuanhiep1232@gmail.com, kyungbaekkim@jnu.ac.kr

***Abstract*—Software Defined Network (SDN) empowers network operators with more flexibility to program their networks. In SDN, dummy switches on the data plane dynamically forward packets based on the rules which are managed by a centralized controller. To apply the rules, switches need to write the rules in its flow table. However, because the size of the flow table is limited, a scalability problem can be an issue. Also, this scalability problem becomes a security issue related to Distributed Denial of Service (DDoS) attacks, especially the resource attack which consumes all flow tables of switches. In this paper, we explore the impact of the resource attack to a SDN network. The resource attack is emulated on the SDN with mininet and OpenDaylight, and the effect of resource attack to the SDN is deeply analyzed in the aspects of delay and bandwidth. Through the evaluation, we highlight the importance of managing the flow tables with the awareness of their size limitation. Also, we discuss solutions which can address the resource attack and their challenges.**

I. Introduction

SDN opens a new approach to design and manage a network. SDN separates the control plane and data plane, which are tightly coupled in a traditional network. But, in SDN, the network devices such as switches and access points are responsible for forwarding packets based on rules come from a centralized controller. The centralized controller acts as a brain which processes routing information and makes decision for configure a network, and it manages the rules for how to handle the packets. The network device has a flow table which contains flow entries which hold the rules. The action of the rules can be drop, forward to the output port, and etc. These rules are dynamically deployed in network devices whenever a new packet comes in. Every time when a network device receives a packet, the device tries to find a flow entry which has the matches to the header of the packet. If the network device does not find any matched flow entry, it asks the controller how to handle the packet. Then the controller installs a flow entry to the device by following the matching rule for the packet. Next time when the network device receives a similar packet, it can process the packet by referring the flow entry.

With this dynamic updates of flow tables of network devices, SDN makes a network more flexible and easier to manage. However, even though SDN provides many benefits to a network, the dynamic feature may lead some security issues. A central controller is a highly valuable target for malicious attackers to compromise a network. A controller is the brain of a network, and if the brain does not work properly the operations of the entire network can be influenced. For example, if a malicious attacker has a control of a controller, he can do everything with the network which is controlled by the compromised controller, such as accessing to confidential servers, preventing legal network traffic, redirecting traffic to incorrect destinations and attacking to a targeted server. Generally obtaining the full control of a controller is not easy, but it is easy for attackers to make a controller very busy and disturbing the operation of the network. DDoS attack is one of the simple attacks to disturb a controller and this kind of attack has been facilitated by user friendly tools such as Stacheldraht.

Especially, in SDN, attackers can exploit the dynamic update feature of network devices, and they makes the resources of network devices overloaded by sending bogus packets. This kind attack is called as resource attack. In SDN, the resource attack can be easily facilitated because of the limitation of flow table size in network devices [1] [2] [3]. A flow table which contains the information for handling packets is the most importance component of a network device. But, its size is limited as few hundreds of flow entries for recent OpenFlow switches [4] [5] . With this size limitation, attackers can easily fill the flow table of network devices with few hundreds of packets. A similar attacking scenario was mentioned in [6]. In this scenario, if an attacker recognizes a network is operated by a controller, he sends the different packet header to make the flow table full. After that, he sends the high load traffics which are not matched with any flow entries, and the operation of the network is disturbed. However, the impact of this resource attack has not been explored in detail. In this paper, we explore the impact of the resource attack to SDN in the aspect of delays and bandwidth. Through a series of extensive emulation of resource attacks, we investigate how the properties of SDN, such as the size of a network and the delay between a controller and a switch, are related to the effect of the resource attack. With the observation of the impact of the resource attack, we discuss some possible solutions such as proactive mode of switches and software cache module for flow entries, and their challenges.

The paper is organized as follows. In section II, we will describe the operation of SDN network, the possible consequences when the flow table reaches the limitation, and then we will describe detail the operation of the resource attack scenario and its possible influences. In section III, we will describe the environment of the emulation of resource attacks and the analysis of results. Section IV discusses the possible solutions to mitigate the resource attack and their challenges. Finally, we conclude the paper in section V.

II. Rationale of Resource Attack in SDN

In this section, we explain the basic SDN operations and the abnormal SDN operations when the flow table reaches its limitation. Also we describe a resource attack scenario that based on the flow table limitation issue.

A. SDN Operation

The architecture of SDN is composed of three layers such as Application layer, Control layer, and Infrastructure layer [7]. Figure 1 depicts the architecture of SDN. The infrastructure layer (or the data plane) includes network element such as switches and access points. The role of network element is to forward a packet to its destination or drop it. The control layer includes a controller which interacts with the network elements on the infrastructure layer through southbound interface. It also provides an interface to applications of the network (northbound interface). The application layer includes applications which control the network via northbound interface based on the policies given by network operator.

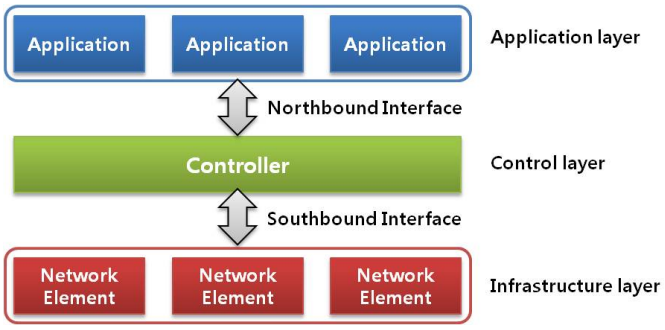


Figure 1: SDN Architecture

Match fields	Priority	Counters	Instructions	Timeouts	Cookie

Figure 2: Flow Entry in a Flow Table

Generally, the northbound interface is implemented as a RESTful API, and the southbound interface implements the OpenFlow Protocol [8]. The OpenFlow Protocol is a protocol for a controller to communicate with network devices for configuring and monitoring them. In the OpenFlow Protocol, each network device has a flow table which contains flow entries. Figure [**?**] describes fields of a flow entry in a flow table. The match fields consist of the information of ingress ports and packet header information, such as source/destination IP/MAC address, Ethernet type and source/destination port, and other metadata which are specified in flow tables of other network devices [9]. When a network device receives a packet, it looks up its flow table and tries to find flow entries which have the matched information of the packet header. If the packet header matches multiple flow entries, the flow entry which has highest priority value is used. The counters field indicates how many times the flow entry is used by increasing its value whenever it is used. The instruction field has a list of actions which indicate how to handle packets such as forwarding packets to output ports, dropping packets, and modifying the payload or the header of the packets. The timeouts field is used for cleaning up flow entries, that is, deleting flow entries from a flow table. The cookie field can be used by controller to filter flow statistic, flow modification and flow detection. This cookie field is not used when processing packets.

Figure 3 illustrates the basic SDN operations for transferring traffic of a flow. Firstly, the source host sends the first packet of a flow to the network device (1). Secondly, the network device looks up matched flow entries. If there is no matched flow entry, the network device sends a request to the controller (2). In the controller, the request will be forwarded to the control application. Thirdly, based on the knowledge of the network, the application makes a decision and tries to install flow entries to all of the network devices along the path of the packet (3). After installing the flow entries, the application sends back the packet to the network device which requests the flow entries as well as to the last network device of the path of the packet. Then the last network device forwards the first packet to the destination host (4). After deploying flow entries in every network devices, the network devices handle the following packets of the flow by referring the instruction field of the deployed flow entry (5).

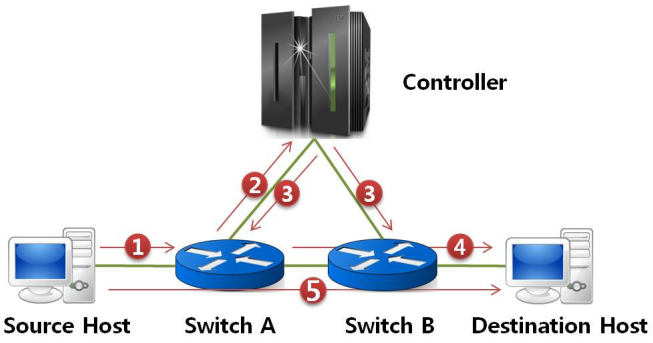
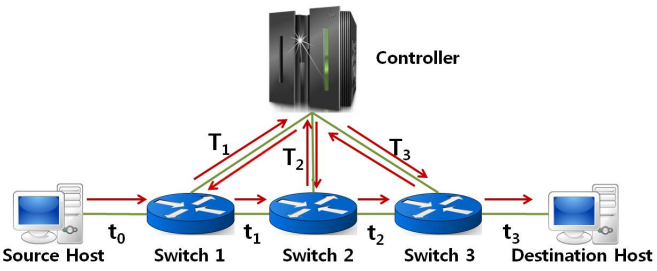


Figure 3: Basic SDN operation of setting up a network flow

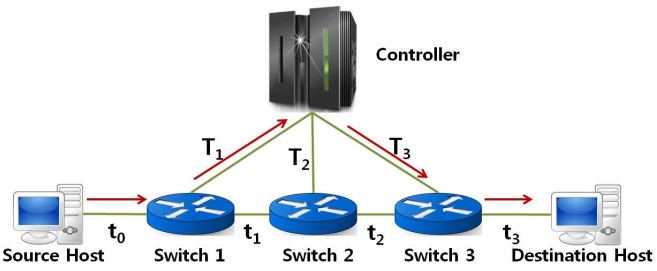
B. Issues of Flow Table

In order to support the correct SDN operation, one of key resources is the flow table of a network device. Because the flow table is updated dynamically and frequently, generally many SDN devices employ the Ternary Content-Addressable Memory (TCAM) as storing the flow table and lookup flow entries for a given packet header. TCAM is a specialized high-speed memory that searches its entire contents in a single clock cycle, but it has some disadvantages; such as high power consumption, high cost, and low utility of ASCI space. That is, increasing the memory space of TCAM can lead to other problems about the price, the energy, and the physical size of network devices. According to this, generally the size of flow table of a network device is not that big. For example, the 5406zl switch can support about 1500 OpenFlow rules or 64000 forwarding entries for standard Ethernet switching [4].

This size issue of a flow table has not been concerned seriously under legacy networks, but in SDN it can be a serious issue. SDN is traffic oriental rather than host oriental,



(a)



(b)

Figure 4: Different procedures of handling a new flow request when all of flow tables are full

which means we may have to install multiple flow entries for each host in the network. Therefore, the number of the flows may be much bigger than the number of hosts in the network. That is, there is a high possibility that a flow table reaches its limitation with normal SDN operations. While the size limitation of a flow table causes a scalability issue, it may also cause a security issue. That is, the size limitation of a flow table may be an attractive attack point for malicious attackers who want to subvert a SDN network. Attackers easily fill up a flow table by sending raw packets with different packet headers. Actually this kind of attack requires simple knowledge of network programming and it can be easily facilitated by using naive traffic generation tools.

When a flow table becomes full, the performance of a SDN flow decreases in aspects of delay and overheads. Let us assume that there is a SDN which is composed of n switches which are connected in a serial formation like Figure 4. In this figure, T_i is the delay between i_{th} network device and the controller and t_i is the delay between i_{th} network device to $(i+1)_{th}$ network device except t_0 (between the source host and the first network device) and t_n (between the last network device and the destination host). In this setting, the normal flow whose matching rules are deployed in all of the flow tables passes only all of the network devices, and its packet delay can be represented as $D_{deployed_flow} = \sum_{i=0}^{n}(t_i)$. On the other hand, the delay of the flow whose matching rules are not deployed in network devices whose flow tables are full depends on the handling procedure of a controller application. Figure 4a and 4b illustrate two different procedure of handling the new request when flow tables of all network devices are full, and show the traffic path of the new flow for each case.

In Figure 4a, the controller application reacts immediately. That is, it tries to deploy new rules on the flow table of the network device which sends a request of a new flow. In this case, the replacement of flow entries happens in the network device. However, if the network is crowded enormously, every network device may need to conduct this replacement of flow entries even though the controller application updates all of the flow tables in the flow path. So, with the conservative controller application, the worst case of the packet delay of a flow over fully occupied network devices can be represented as $D_{undeployed-flow} = t_0 + \sum_{i=1}^{n}(2T_i + t_i)$. In this worst case, the controller application needs to handle n requests for a packet of an undeployed flow.

The other procedure of handling undeployed flows is illustrated in Figure 4b. In this case, the controller application does not replace flow entries immediately but forward packets of undeployed flows by itself. That is, the controller application knows the path of an undeployed flow and forwards its packets to the last network device of the path directly. The controller application poses optimistic characteristic and waits for expiration of some flow entries. The delay of a undeployed flow with the optimistic controller application is represented as $D_{undeployed-flow} = t_0 + T_1 + T_n + t_n$, and the controller application handles 2 requests for a packet of an undeployed flow.

C. Resource Attack related to flow tables

In the previous section, it is shown that the issues of flow tables can lead to the performance issue of a controller as well as flows. Also, malicious attackers may exploit the issue of flow tables to subvert a SDN. Attackers simply make flow tables full with bogus flows by continuously sending the packets which have slightly different header information. Under this simple attack, the flow tables are always full and the legitimate flows suffer from addition delays for replacing flow entries. Also the enormous additional flow entry requests exhaust the computing resource of the controller and its applications. Moreover, attackers may make attacking tools more sophisticate as generating high load bogus flows which may not match any flow entry in many network devices. Then, the impact of the resource attack becomes more serious to the performance of SDN.

III. Evaluation of Resource Attack

In this section we describe details of the environment for evaluating the impact of resource attacks and analyze the evaluation results. Firstly, we evaluate the delay and the bandwidth of deployed flows and undeployed flows when the flow table size reaches its limitation. Secondly, we evaluate the percentage of packet drops in the case of simultaneously transferring multiple flows under a resource attack.

In our evaluation environment, we use OpenDayLight [10] as a centralized controller and Mininet [11] for emulating network devices and hosts. Mininet and OpenDayLight are both installed on a single virtual machine operated with Ubuntu 13.04. The virtual machine has 3GB memory and 2 3.4GHz CPUs. Figure 5 illustrates the network topology of our evaluation environment. The emulated network is composed of a controller, n switches and $2m$ hosts. All switches are indexed from 1 to n. Every switch has emulated Ethernet interfaces for connecting to its adjoining switches. Hosts are indexed from 1

to $2m$ and divided into two groups. Hosts indexed from 1 to m belong to the first group and other hosts belong to the second group. All hosts of the first group have a link to connect with the first switch and all hosts of the second group link to the last switch. In our experiment, a host of the first group sends packets to only one host of the second group. That is, i_{th} host sends packets to $(i+m)_{th}$ host.

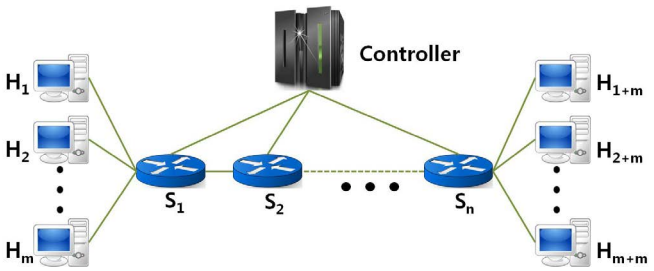


Figure 5: Network Topology

In our first experiment, we emulate a network topology which consists of 20 hosts and various numbers of switches. The number of switches is varied from 2 to 6. In this experiment setting, the controller-switch delay (delay between the controller and a switch) is set to 100ms and the delay between two switches is 30ms. We limit the number of flow entries for all switches. This can be done by using OpenV vSwitch commands. The flow table size limitation is set to support only 5 flow entries. After setting the size limitation of all of flow tables, we successively send packets from first groups host to second groups host by using the ping command. Then, we record the delay of the first packet and the average delay of the residual packets for each flow. In this experiment, we have 10 flows. For the first five flows, the SDN controller correctly installs flow entries to the switches after receiving first packet. Therefore, the residual packets are forwarded through the switches normally. But for the last five flows, the matched flow entries cannot be deployed on switches because the flow tables of switches are full. So, the residual packets and first packet have to go through the controller. The path of the packets can be one of two cases that was mentioned in Section II-B. We call the first five flows as deployed flows and the last five flows as undeployed flows. We also measure bandwidth of flows by using iperf.

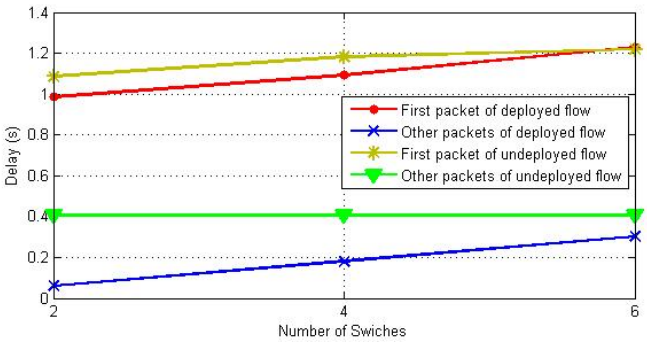


Figure 6: Relation between delay and number of switches

Figure 6 shows the relationship between delay of flows and number of switches. We observed that the first packet delay is much higher than average delay of the residual packets. It is because of the delay of handling the first packet in the controller. When the first request of a flow comes to the controller, the controller needs to process the flow such as looking the destination and finding the path to route packets to the destination, then the controller tries to install the flow entries to all of the switches related to the path of the flow. The first packet delay slightly increases along with the number of switches, because the controller has to install flow entries to all switches in the flow path. While the delay of the first packet of undeployed flows is similar to deployed flows, the average delay of other packets of undeployed flows is quiet different to deployed flows. We observed that the average delay of other packets of deployed flow increases with the number of switches but the average delay of other packets of undeployed flows exhibits a constant value. After the first packet of a deployed flow, the flow entries are installed in the switches and the packets just pass through the switches without asking the controller. Therefore, for deployed flows, when number of switches increases, the delay also increases. On the other hand, the undeployed flows pass through the controller rather than the switch network, the average delay does not changes even though the number of switches change.

Figure 7 shows the relationship between delay of flows and the delay between the controller and switches. In this setting, the number of switches is set to 4 and the controller-switch delay varies from 50ms to 300ms. The delay of the first packet increases as the controller-switch delay increases. Also, similar to Figure 6, deployed flows and undeployed flows have similar delay of the first packet. However, unlike the previous result, the delay of other packets of undeployed flows increases along with the controller-switch delay and the dealy of other packets of deployed flows has a constant value. This consolidates our assumption that packets of undeployed pass through first switch, controller and last switch and then go to its destination.

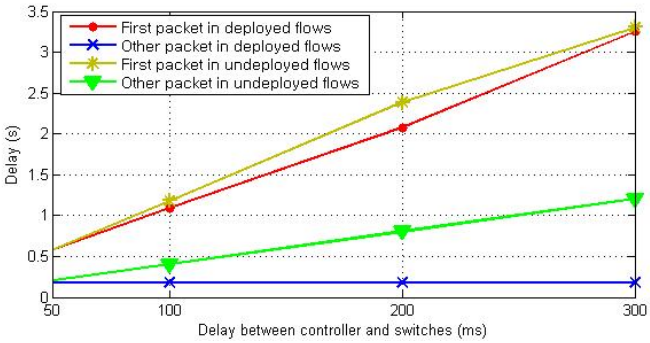


Figure 7: Relation between delay and delay between controller and switches

Figure 8 and 9 show the relationship between bandwidth and the number of switches and the relationship between bandwidth and the controller-switch delay, respectively. At first, we observed that the bandwidth of undeployed flows is much smaller than the bandwidth of deployed flows. There are two main factors which suppress the bandwidth of undeployed flows; the workloads of the controller and the controller-switch

delay. Because the packets of undeployed flows pass through the controller, the bandwidth is bounded to the computing power of the controller. In Figure 8, the bandwidth of undeployed flows is less than 10 MBits/s while the bandwidth of deployed flow achieves more than 30 MBits/s. In this evaluation the controller runs with Mininet, the computing resource of the controller is low and it provides low bandwidth for undeployed flows. In Figure 9 we observed that the bandwidth of undeployed flows decreases as the controller-switch delay increases.

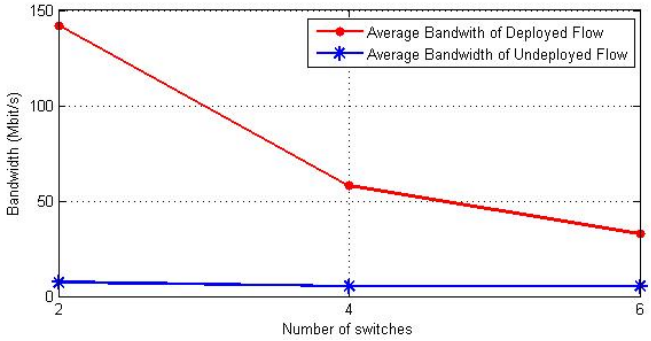


Figure 8: Relation between bandwidth and number of switches

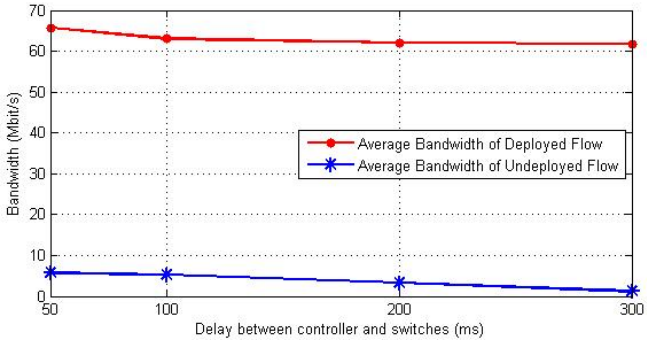


Figure 9: Relation between bandwidth and delay between controller and switches

In the previous experiments, the packets of flows are inserted in the network sequentially. In order to evaluate the impact of resource attack in more realistic environment, we run multiple flows in the same time and measure the bandwidth and packet drop ratio of undeployed flows and deployed flows. In this evaluation, we use a network topology consisting of 70 hosts and 4 switches. The controller-switch delay is set to 100ms and set the flow table size limitation as 17. We have 17 deployed flows and 18 undeployed flows. To run simultaneous undeployed flows or deployed flows, each host in the second group initiates an UDP server and each host in the first group runs an UDP client simultaneously. The UDP client sends a 1000 byte packet in every 1ms and every packet has its ID and the timestamp. Clients and servers store the packet logs and these logs are used to measure packet drop ratio and bandwidth. In this evaluation, the number of simultaneous deployed flow and undeployed flows are varied from 5 to 15.

Figure 10 represents the effect of number of simultaneous flows to bandwidth and packet drop ratio. As the number of simultaneous undeployed flows increases, the packet drop ratio of deplyed flow and undeployed flow increase exponentially and the their bandwidth decrease linearly. The packet drop ratio of undeployed flow is more than 80% if there are 15 simultaneous undeployed flows. The average bandwidth of undeployed flow is less than 4Mbit/s while deployd flow achieves more than 20Mbit/s. The diferences between deployed flow and undeployed flow are because the controller spends more resouces for processing the packet than switch. The controller does the complex processing such as calculating the appropriate route, managing topology rather than just checking flow table. In additional, the controller is getting more overloaded as the number of flows increases.

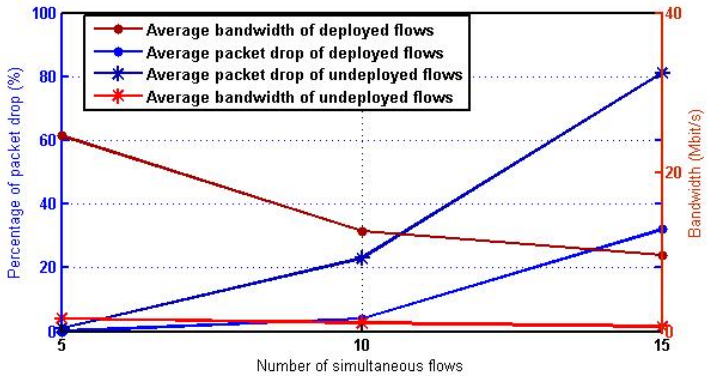


Figure 10: Relation between bandwidth, packet drop and number of flows

IV. Discussion

Through the emulation of resource attacks with OpenDayLight controller and its application, we observed that the packets of an undeployed flow pass through the controller and they are directly forwarded to the last network device. In this case, the simple resource attack to flow tables may significantly degrade the performance of the controller because all of the packets of undeployed flows pass through the controller and the computing resource of the controller is exhausted by handling these packes for undeployed flows. Also, the bandwidth of undeployed flows is significantly degraded. Moreover, if packets pass through the controller, it causes security issues. For example, let us assume that there is a firewall or other middleboxes between within a network. If flow tables of some network devices are full and packets are bypassed the network, the network cannot utilize the functionalities of middleboxes correctly, then sometimes it allows illegal accesses to confidential servers.

According to this analysis, we conclude that the issues of flow table, especially the size limitation of flow table, should be considered very carefully during building a SDN network and developing controller applications. Here we provide some considerable advices related to mitigate the impact of the resource attack to flow tables. First, the simplest solution to avoid the resource attack is setting network devices in the proactive forwarding mode rather than the reactive forwarding mode. In the proactive forwarding mode, the network flow

entries are designed and configures in advance that network devices are used. The SDN control application does not install flow entries dynamically, but mainly monitor the status of network devices and suggest some recommendation of flow entries to network administrators. However, because the network configuration is designed and deployed by human operators, scalability and human factor (errors) can be issued. Also, this mode is contrary to the basic concept of SDN which programs a network dynamically.

Secondly, a SDN controller application should consider the size limitation of network devices and avoid unexpected behaviors. That is, the controller application should handle the case that the flow table of a device is full, and define the detail handling procedure which supports the purpose of the network correctly. As we discussed earlier, if a firewall is located in the middle of a network and every packet supposes to pass through the firewall, the controller application should guarantee that every flow does not bypass the firewall even though the situation of full flow tables happens. To design the detail handling procedure, the controller should have the current knowledge of all network devices such as the size of flow table and the usage of flow entries.

To ensure the traffic go through network devices, a controller application should replace the required flow entries with an old flow entry of a fully occupied flow table in right time. The key issue of the replacement is how to choose the old flow which is going to be replaced. A simple replacement policy is using the timeout filed of flow entries. That is, the controller application chooses the flow entry which is going to be expired at the earliest. For better replacement policies, the controller application may use multiple parameters such as number of packets of a flow entry, generation date of a flow entry and utilization of a flow entry.

Thirdly, a SDN controller may require an intermediate module which stores the flow entries temporarily and conducts the replacement of flow entries. Monitoring every network devices and preprocessing the replacement procedure becomes additional overheads of a controller which is the most valuable resource in SDN. To relieve these overheads, it is better to separate the functionality of handling exceptional cases from the basic operations of a controller. Because of this separation, this intermediate module has additional functionality of detecting resource attacks to support better replacement policies. That is, a controller does not need to replace an attack flow with a legitimate flow. It may be possible to recognizing that a resource attack is being facilitated by checking the growth speed of flow entries in SDN. Whenever the controller recognizes that the network is under attack, it operates the network more conservatively.

V. Conclusion

In this paper, we have examined the impact of resource attacks which exploit the size limitation of a flow table to the performance of SDN. Our evaluation shows that the performance of a controller is significantly degraded under resource attacks if a controller application does not define the detail handling procedures for undeployed flows. Moreover, the size limitation issue may cause the unexpected behaviors of a controller, which create additional security problems. . Because the resource attack can be easily facilitated with basic knowledges of networking and simple networking tools, the countermeasure against the resource attack should be considered more carefully during designing controller applications and configuring SDN environments. As a future work, we are working on designing and developing a flow table manager with a smart replacement algorithm. The flow table manager should classify the flow entries into attack flows and legitimate flows and apply replacement policies which minimize the impact of the resource attack.

VI. Acknowledgement

This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2014R1A1A1007734).

References

[1] D. Kreutz, F. M. V. Ramos, and P. Verssimo, "Towards secure and dependable software-defined networks." in *HotSDN*, N. Foster and R. Sherwood, Eds. ACM, 2013, pp. 55–60. [Online]. Available: http://dblp.uni-trier.de/db/conf/sigcomm/hotsdn2013.html

[2] D. Li, X. Hong, and J. Bowman, "Evaluation of security vulnerabilities by using protogeni as a launchpad." in *GLOBECOM*. IEEE, 2011, pp. 1–6. [Online]. Available: http://dblp.uni-trier.de/db/conf/globecom/globecom2011.html

[3] K. Benton, L. J. Camp, and C. Small, "Openflow vulnerability assessment." in *HotSDN*, N. Foster and R. Sherwood, Eds. ACM, 2013, pp. 151–152. [Online]. Available: http://dblp.uni-trier.de/db/conf/sigcomm/hotsdn2013.html

[4] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalag, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," in *In ACM SIGCOMM*, 2011.

[5] N. P. Katta, J. Rexford, and D. Walker, "Incremental consistent updates." in *HotSDN*, N. Foster and R. Sherwood, Eds. ACM, 2013, pp. 49–54. [Online]. Available: http://dblp.uni-trier.de/db/conf/sigcomm/hotsdn2013.html

[6] S. Shin and G. Gu, "Attacking software-defined networks: A first feasibility study," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13. New York, NY, USA: ACM, 2013, pp. 165–166. [Online]. Available: http://doi.acm.org/10.1145/2491185.2491220

[7] "Sdn architecture," june 2014, accessed: 2014-09-12. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[9] "Openflow switch specification," oct 2013, accessed: 2014-09-12. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf

[10] Accessed: 2014-09-12. [Online]. Available: http://www.opendaylight.org/

[11] Accessed: 2014-09-12. [Online]. Available: http://mininet.org/