Attacking Software-Defined Networks: A First Feasibility Study

[Extended Abstract]

Seungwon Shin
SUCCESS Lab, Texas A&M University
seungwon.shin@neo.tamu.edu

Guofei Gu
SUCCESS Lab, Texas A&M University
guofei@cse.tamu.edu

ABSTRACT

In this paper, for the first time we show a new attack to fingerprint SDN networks and further launch efficient resource consumption attacks. This attack demonstrates that SDN brings new security issues that may not be ignored. We provide the first feasibility study of such attack and hope to stimulate further studies in SDN security research.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

Keywords

Software Defined Networking, Network Security, Attack

1. INTRODUCTION

In this work, we demonstrate an effective and efficient attack against software-defined networks with the knowledge of some basic characteristics of the SDN technology. Essentially, since the control plane is separated from the data plane in a SDN network, the data plane will typically ask the control plane to obtain flow rules when the data plane sees new network packets that it does not know how to handle. By exploiting this key property, our proposed attack can first fingerprint whether a given network uses SDN/OpenFlow switches and then generate specifically crafted flow requests from the data plane to the control plane. This has two effects: (i) it can make the (logically centralized single-point) control plane hard to handle all requests, i.e., **control plane resource consumption or Denial-of-Service (DoS) attack**; (ii) the generated fake flow requests can produce many useless flow rules that need to be held by the data plane, thus making the data plane hard to store flow rules for normal network flows (**data plane resource consumption or DoS attack**). To demonstrate the feasibility of such attack, we create a new SDN network scanning prototype tool (named as SDN SCANNER) to remotely fingerprint networks that deploy SDN, and this method can be easily implemented by modifying existing network scanning tools (e.g., ICMP scanning and TCP SYN scanning).

HotSDN'13, August 16, 2013, Hong Kong, China.
ACM 978-1-4503-2178-5/13/08.

2. MOTIVATION AND ATTACK METHOD

2.1 Motivation

In a SDN environment, the control plane can dynamically enforce flow rules when the data plane requires, and it enables us to control the network efficiently. However, this kind of reactive-mode control can cause serious problem when there are too many requests from the data plane to the control plane. If the data plane receives many requests in a short time period, it can flood the messages to the control plane. In addition, a flow table in the data plane can also be flooded by the rules for handling the requests.

2.2 Fingerprinting a SDN network

If a client sends packets to a SDN network, this client will observe different response times, because the flow setup time can be added in the case of New-Flow (i.e., no flow rule for handling packets in the data plane) compared with the case of Existing-Flow (i.e., there is a flow rule for handling packets in the data plane). To describe this more clearly, we simply formalize the response time that is observed at a client side. First, we define the response time for the Existing-Flow case as α, and the additional flow setup time as β. In addition, for brevity, we define the response time for the case of New-Flow and Existing-Flow as $T1$ and $T2$ respectively, and they can be represented as follows.
$T1$ (w/o flow rule in the data plane) $= \alpha + \beta$
$T2$ (w/ flow rule in the data plane) $= \alpha$

In this case, if an attacker can clearly differentiate $T1$ from $T2$, he/she can fingerprint a SDN network. However, an attacker will still face two challenges: (i) how to collect $T1$ and $T2$ values, and (ii) how to know whether $T1$ values are different from $T2$ when considering random noises.

SDN Scanner: The first challenge can be addressed by our new network scanning method, *header field change scanning*, which scans networks as changing network header fields. When SDN SCANNER collects $T1$ and $T2$ values, it follows the following steps. First, it sends two (or more) specifically crafted packets to a target network and records the response time of each packet. At this time, SDN SCANNER considers the response time for the first packet could represent $T1$, and the time for the second packet shows $T2$. And, SDN SCANNER repeat this operation by changing a

field of the packet header. Finally, SDN SCANNER collects $T1$ and $T2$ for each different header field. This operation is presented in Figure 1.

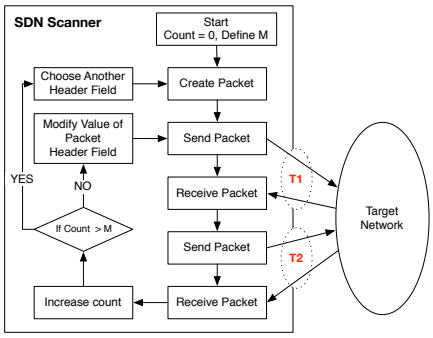

Figure 1: Simplified function diagram of SDN SCANNER

Statistical Testing for Two Sample Sets: Once an attacker collects samples of $T1$ and $T2$ using SDN SCANNER, he/she now faces the second challenge, which can be solved by employing statistical testing methods, such as *t-test* [2]. This method simply tests whether two sample sets are significantly different from each other or not with a high confidence. This test just requires the mean and standard deviation values of each sample that can be easily obtained, and the test method is pretty simple. Of course, an attacker can easily use more advanced statistics or machine learning techniques to improve the accuracy.

2.3 Launching DoS attacks to a SDN network

If an attacker runs SDN SCANNER and collects network information, he/she can investigate whether a target network is using SDN or not through a simple statistical testing method. If the test results show that a target network is likely to use SDN, the attacker will further conduct the resource consumption attack. Since the attacker already knows the condition of the flow rule for the target network (with the help of SDN SCANNER), now he/she just needs to send network packets to consume SDN resources of the target network.

3. EVALUATION

In current network situation, it is very hard to collect this information from the Internet, because SDN is not widely deployed to many networks yet (but we believe that SDN will be employed to many networks soon). Therefore, we have decided to use other measurement results to estimate $T1$ and $T2$ values.

Estimating $T2$: We send 20 `ping` packets to 28 different real-world networks (we call them target networks) to collect $T2$ values, and we collect the response times from the second packets (i.e., ignore the response time for the first packet) to avoid any possibility of including flow setup time of a SDN network. We send `ping` packets from a state in U.S.A., and the various locations of the target networks are distributed in the same state, in different states (of the same continent), and in different continents.

Estimating $T1$ Values: It is very hard to get the information of $T1$, thus we decide to estimate $T1$ by adding flow setup time to $T2$ values. With the help of the previous work [4], we can get the information of flow setup time for three different control plane cases: NOX, Beacon, and Maestro.

Fingerprinting Result: We apply *t-test* [2] to collected $T2$ and estimated $T1$ samples to figure out if $T1$ is significantly different from $T2$, and we find that SDN SCANNER can fingerprint 24 networks out of 28 cases (i.e., the fingerprinting rate is 85.7%).

DoS Attack Result: We have set up a test environment to understand whether the proposed DoS attack is successful or not, and the environment consists of a single OpenFlow switch, a controller, and two hosts for network communications. We use the software based OpenFlow switch implementation for the OpenFlow switch [3], and it is installed on an independent Linux host, and we set the maximum flow rules for this switch as 1,500, which is the same configuration for HP 5406zl switch [1]. Figure 2 shows the time and bandwidth that are required for a DoS attack to consume resources of the control plane and the data plane.

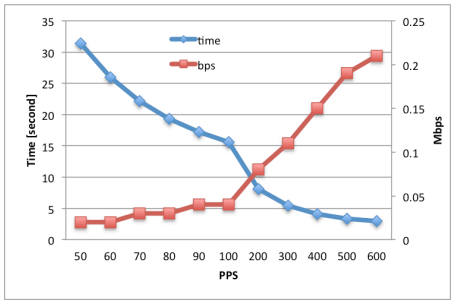

Figure 2: Required attack time and network bandwidth for DoS attacks

4. CONCLUSION AND FUTURE WORK

In this paper, we introduce a new fingerprinting attack against SDN networks, and we also show its feasibility with real world experimental data. To the best of our knowledge, the proposed attack scenario is the first realistic attack case to a SDN network that can be conducted by a remote attacker, and this attack could significantly degrade the performance of a SDN network without requiring high performance or high capacity devices. In our future work, we will set up a more realistic SDN network environment for our evaluation, further improve SDN SCANNER, and design new defense solutions.

5. REFERENCES

[1] A. Curtis, J. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee. Devoflow: Scaling flow management for high-performance networks. In *Proceedings of ACM SIGCOMM*, 2011.

[2] J. Fisher Box. Guinness, gosset, fisher, and small samples. In *Statistical Science*, 1987.

[3] OpenFlow.org. Openflow switching reference system. `http://www.openflow.org/wp/downloads/`.

[4] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood. On controller performance in software-defined networks. In *Proceedings of HotICE*, 2012.