International Journal of Security and Its Applications
Vol.8, No.5 (2014), pp.119-130
http://dx.doi.org/10.14257/ijsia.2014.8.5.12

Traffic-based Malicious Switch Detection in SDN

Xiaodong Du[1], Ming-Zhong Wang[1], Xiaoping Zhang[2*] and Liehuang Zhu[1]

[1]Beijing Engineering Research Center of Massive Language Information Processing and Cloud Computing Application
School of Computer Science Beijing Institute of Technology
[2]National Key Lab of Vehicular Transmission
China North Vehicle Research Institute
{ronniedxd, wangmz, liehuangz}@bit.edu.cn, zxp_cnvri@163.com

Abstract

In Software Defined Networking (SDN) architecture, the control plane is separated from the data plane. On one hand, OpenFlow switches can only store and forward packets, which leaves all decisions to be made by the controller. On the other hand, the controller has a global view over the SDN. But if any switch is captured by an adversary, it may mislead the controller to make inaccurate decisions which may have terrible influences on the overall networks. In this paper, we elaborate on these problems and propose methods to detect malicious OpenFlow switches. We set a threshold value of the traffic-flows across an OpenFlow switch. If the switch's current traffic-flows exceed the threshold value, the controller has reasons to believe that this switch is suspicious and may monitor it intensively. Another scheme is to add a third-party server to accept users' report to warn the controller. In SDN, the controller cannot communicate with users directly, and sometimes users need to feed back their experience to the controller to help improve the SDN. In this case, it is necessary to set a third-party server in SDN to act as a middle role. These two schemes help to detect malicious switches. The controller can analyze the flow table of the suspicious switch and identify whether it is really malicious before isolating it.

Keywords: *SDN, OpenFlow, malicious switch, detection*

1. Introduction

In traditional networks, the control plane and the data plane are in routers. Each router makes independent decisions based on its routing table and maintains its routing table autonomously, or it is manually set by administrators. Router operation logic is built into hardware. New functionalities or updates need to be performed on hardware. The complexity of today's networks makes it very difficult to apply a consistent set of access, security, QoS, and other policies. Thus, SDN is proposed to replace traditional networks. Figure 1 shows the standard architecture of SDN.

* Correspondence author

ISSN: 1738-9976 IJSIA

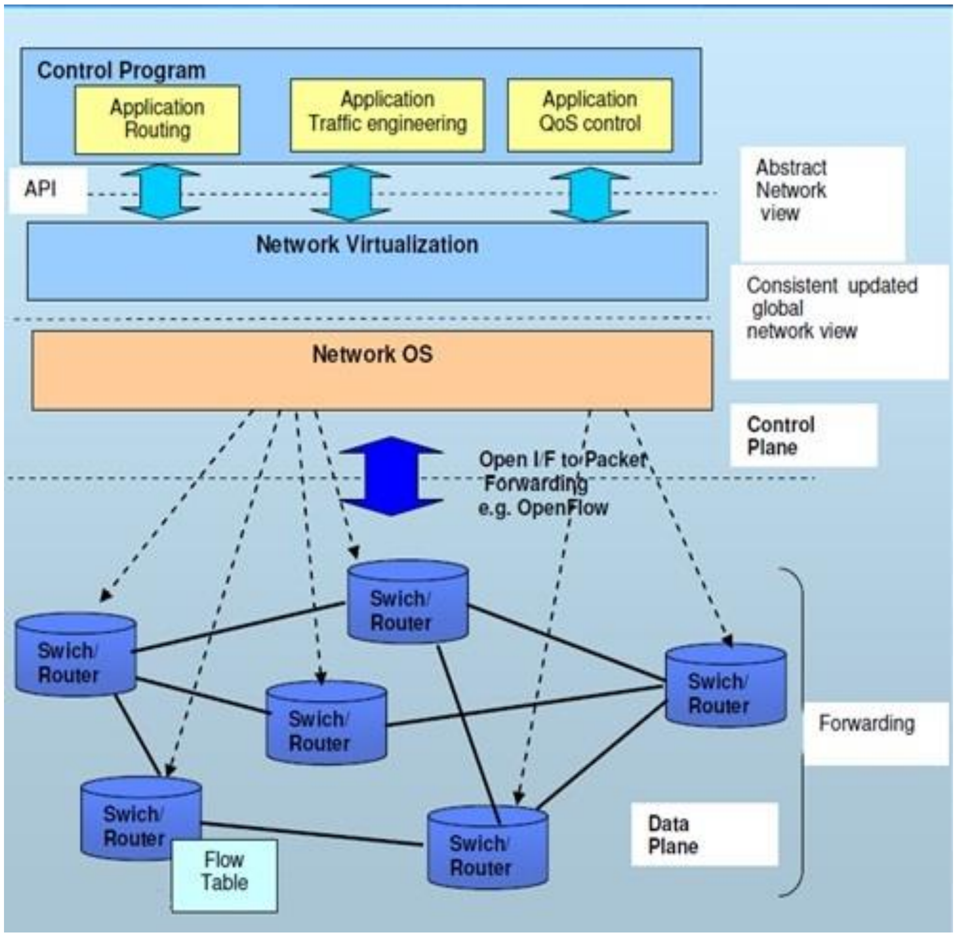

Figure 1. The Basic Architecture of SDN

SDN is a more centralized approach, in which the controller communicates with switches via OpenFlow protocol. The controller is just like a God role in SDN, and each switch works under the instructions from it. Switches only have storing and forwarding functions. However, the controller knows all the information of the networks, including topological structure, the bandwidth between neighboring switches and so on.

If one switch is malicious, it will send inaccurate messages to the controller. For instance, a malicious switch may inform the controller that its processing power and its bandwidth are very huge, which may mislead the controller to assign more flows to the switch. The malicious switch may lead to a large amount of packet loss, which will have a terrible effect on the SDN's performance. In following sections, we illustrate these problems and propose our schemes to solve it.

The remainder of this paper is organized as follows. Section 2 elaborates an example and the motivation. Section 3 introduces and explains the proposed scheme. Section 4 provides performance analyze of the scheme. Section 5 provides a survey on related work and Section 6 concludes.

2. Motivation and Example

A typical SDN topology is shown in Figure 2.

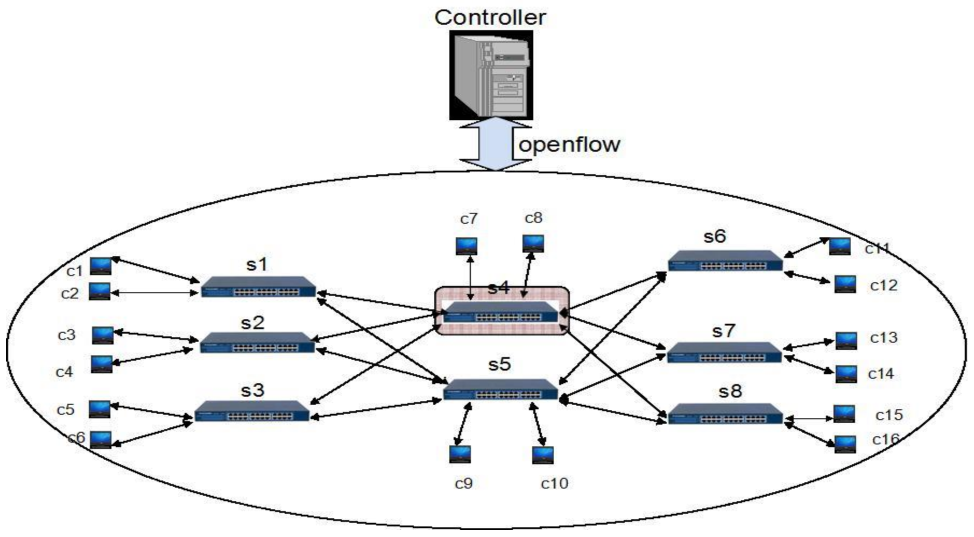

Figure 2. A Typical SDN Topology

There are 8 switches named from s1 to s8, and 16 computers named from c1 to c16. The controller can reach all switches in this network and it communicates with every switch using OpenFlow protocol via a secure channel, typically SSL or TLS. Switch s4 is surrounded by a pink area, which represents s4 is a malicious switch. Other switches are normal switches and function well.

Assume that the bandwidths of s1-s4, s2-s4, s3-s4, s4-s6, s4-s7, s4-s8 are 2Mb/s, the bandwidths of s1-s5, s2-s5, s3-s5, s5-s6, s5-s7, s5-s8 are 4Mb/s. If c1 wants to communicate with c16, the controller tends to assign the flow across s1-s5-s8. On traditional condition, if all switches are normal, the networks will work well. But when s4 is captured by an adversary and turns to be a malicious one, s4 sends messages to the controller to inform the controller that the bandwidths of s1-s4, s2-s4, s3-s4, s4-s6, s4-s7, s4-s8 are 10Mb/s. Next, when new packets arrive at s1 from c1, the controller prefers to forward packets through s4 because it believes that the bandwidth of s1-s4 is wider than s1-s5. This will lead to a lot of packet loss and make a terrible influence on the communication quality of the networks.

In this example, s4 is a malicious switch, which the controller and all the other normal switches do not realize. We just give a simple example about the threat which will be caused by a malicious switch. In fact, a malicious switch can make worse damages to SDN. Therefore, we should take some active steps to solve it.

3. Proposed Schemes

In this section, we elaborate on solutions that can help the controller to detect malicious switches.

3.1. Threshold Value Control

We need to maintain a table in the controller platform. This table is mainly used for storing the maximum traffic-flows of each switch in Open Flow networks. The threshold table, MTTV (Maximum Traffic-flows Threshold Value), is designed as shown in Table 1.

Table 1. MTTV Table

Switch Name	IP	MAC	Threshold
S1	192.168.10.101	05-16-DC-59-C2-34	3Mb/s
S2	192.168.10.102	FF-EC-DC-12-43-33	4Mb/s
......	......	......	......
Sn	192.168.10.187	DC-FE-32-29-28-37	3Mb/s

The Table is maintained by the controller which has a global view of Open Flow networks. The *IP* and *MAC* fields aim to locate the unique switch in networks. Each switch has a *Threshold* field. The controller finds out the threshold value of each switch's maximum traffic-flows by learning from its working history.

In the example we have discussed in Section 2, when the network is initialized, the controller will communicate with switches frequently. The controller needs to acquire how many switches are there in this network and the details of these switches including the maximum traffic-flows threshold values. An example MTTV table in the controller towards Figure 2 is illustrated in Table 2.

Table 2. The MTTV Table in the Controller in the Network of Figure 2

Switch Name	IP	MAC	Threshold
S1	192.168.10.101	05-16-DC-59-C2-34	3Mb/s
S2	192.168.10.102	FF-EC-DC-12-43-33	4Mb/s
S3	192.168.10.103	56-41-D8-F7-21-AB	5Mb/s
S4	10.108.14.213	DC-FE-32-29-28-37	7Mb/s
S5	10.108.14.214	BB-D5-A3-67-99-4B	11Mb/s
S6	202.218.135.4	88-12-ED-A2-78-DD	3Mb/s
S7	202.218.135.5	87-23-AC-3B-67-33	3Mb/s
S8	202.218.135.6	66-BA-5E-2F-1A-76	3Mb/s

The controller knows more than just the maximum traffic-flows threshold values. Because the controller is responsible for building the topology of SDN, it must know all bandwidths between every two switches. The information will be maintained by the Topology Manager in the controller. An example of the topology and bandwidths of {s1, s2, s3, s4, s5, s6, s7, s8} is shown in Figure 3.

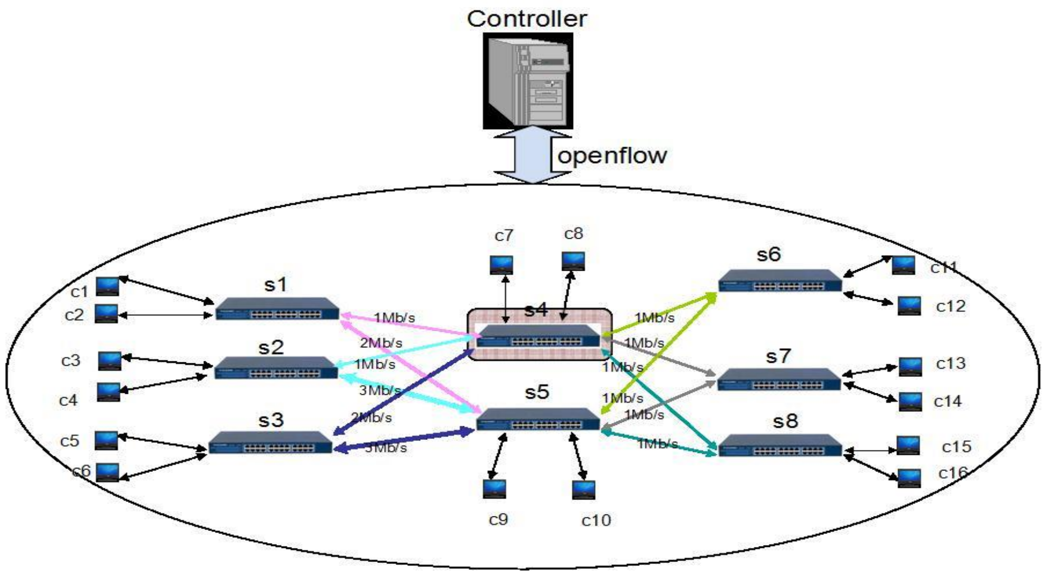

Figure 3. The Topology with Bandwidths

When c1 wants to communicate with c16, there are two choices for the controller to assign this flow. One choice is s1-s4-s8, and the other choice is s1-s5-s8. By observing Figure 3, we will discover that to assign more flows to s1-s5-s8 is a better choice because the bandwidth between s1 and s5 is bigger than that between s1 and s4.

In case that s4 is captured by an adversary and becomes malicious, the adversary tries to lower the quality of service. It will manipulate s4 to mislead the controller. s4 may generate a message and then send it to the controller to inform that the bandwidths of s1-s4, s2-s4, s3-s4, s4-s6, s4-s7, s4-s8 are 10Mb/s, which is far bigger than the fact. This message will convince the controller that assigning more flows to s4 is a wise choice. Thereafter, when new flows arrive at s1 from c1, the controller will tell s1 to forward the flows to s4 rather than s5. This decision is inappropriate because 10Mb/s is far bigger than their real bandwidths and will lead to a lot of packet loss in the network. However, the controller does not realize it.

In this case, the MTTV table will embody its value. Even though the controller does not know whether s4 is telling lies, the controller can refer to the MTTV table when it finds out that the traffic-flows through s4 surge. If the traffic-flows through s4 exceed 7Mb/s, which indicates that s4 does not obtain the capacity to process so many flows, it is hard to avoid lots of packet loss. By checking the MTTV table, the controller may find s4 suspicious easily. When the controller realize some switches (s4 in this case) is suspicious, the controller can inspect the switch and investigate it further. After making sure s4 is a malicious one, the controller can isolate it from the network.

3.2 Third-party Server

We cannot detect all malicious switches totally relying on scheme proposed in Section 3.1. It is necessary for us to design a complementary scheme.

In SDN architecture, users are not connected with the controller directly, and even cannot communicate with the controller. So, the controller does not know what users’ experience are. In this scheme, we add a third-party server to help the controller exchange information with users. This third-party server is mainly used for gathering users’ experience and feedbacks, so we call it UFS (Users Feedback Server). Figure 4 is our design of adding a UFS between the controller and users.

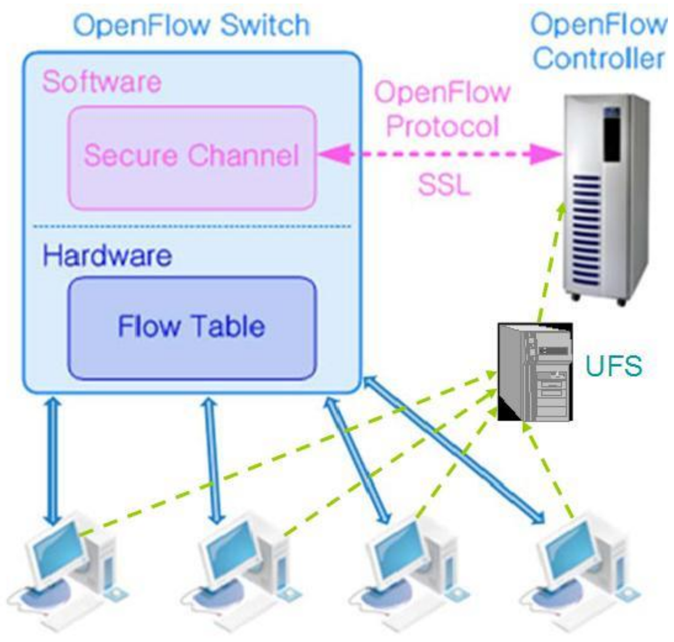

Figure 4. SDN with UFS

In Figure 4, the UFS must be a trusted third party. When users find that it is not easy to connect to the network or there is no response to the traffic they launched, they guess that there are some problems with the route or the network. They can send this information to the UFS. The UFS receives all these feedback information from users and stores them. After a certain period of time, the UFS will send all these information to the controller. The controller learns about the users' experience with the network. By this means, the controller can find some suspicious paths and suspicious switches. In the next moment, the controller can investigate these suspicious paths and switches to identify them. The data structure of the message which is sent by the users is shown is Table 3.

Table 3. Data Structure of the Message

HostName	SrcIP	SrcMAC	DesIP	DesMAC

The field *HostName* refers to the name of a specific computer which is located by the source IP address (*SrcIP*) and the source MAC address (*SrcMAC*). *DesIP* refers to destination computer's IP address and *DesMAC* refers to destination computer's MAC address. With this information in the message, we can point out the source node and destination node of the traffic jam.

With UFS, the typical SDN structure shown in Figure 2 is transformed into Figure 5.

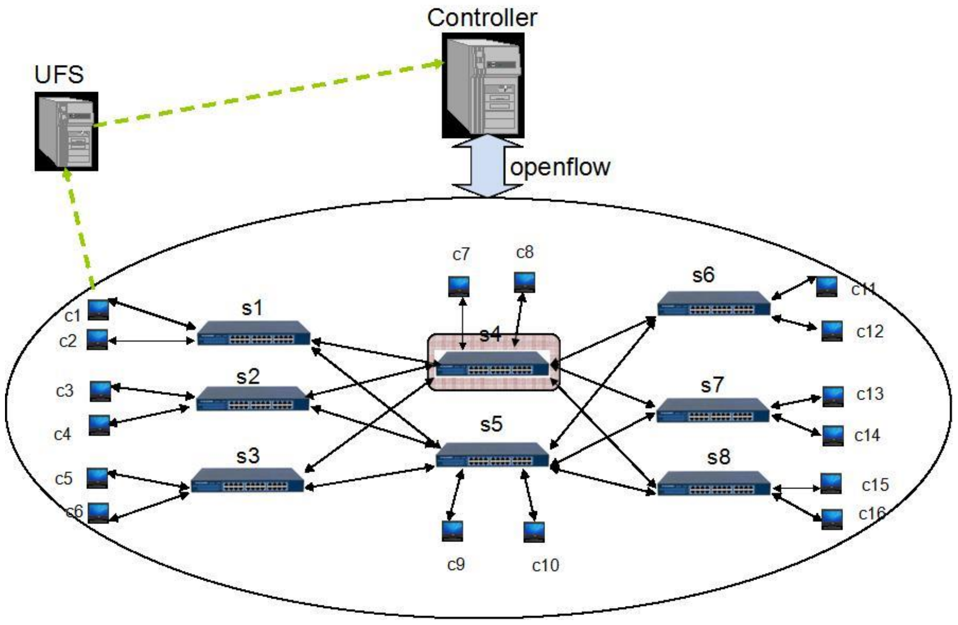

Figure 5. The Integration of SDN in Figure 2 and UFS

In Figure 5, all computers can access the UFS through a web interface. When c1 wants to communicate with c16 and the path of packets flow is c1-s1-s4-s8-c16, the packets sent by c1 will get lost in s4 because s4 is a malicious switch. After c1 realizes the packet loss in communication with c16, c1 generates a message containing the following information: c1's IP and MAC address, c16's IP and MAC address. When this message arrives at UFS, UFS stores it in a Table. The content of the Table is shown in Table 4.

Table 4. The Content of the UFS's Table

HostName	SrcIP	SrcMAC	DesIP	DesMAC
c1	56.23.45.21	EE-23-FC-D2-44-65	228.216.192.36	78-DC-4E-FC-32-B1
......	......	......	......	......

The UFS maintains this Table to store all the communication problems reported by users. After a certain time, the UFS sends the content of the table to the controller to let it know that there are problems between these node pairs. In this case, the controller may find that the flow path from c1 to c16 is obstructed. There are probably suspicious switches in the network. Then the controller could monitor and investigate all the passing traffic between c1 and c16 to detect malicious switches. After analysis, the controller can identify s4 as a malicious switch.

This scheme in this section is a complement to the scheme in Section 3.1. Both schemes can be combined to achieve better effects.

4. Performance Analysis

In this section, we will analyze and compare the performance of a SDN with and without the Threshold Value Control.

If there is a malicious switch in SDN, the inward traffic-flows of the malicious switch along with time will be like Figure 6.

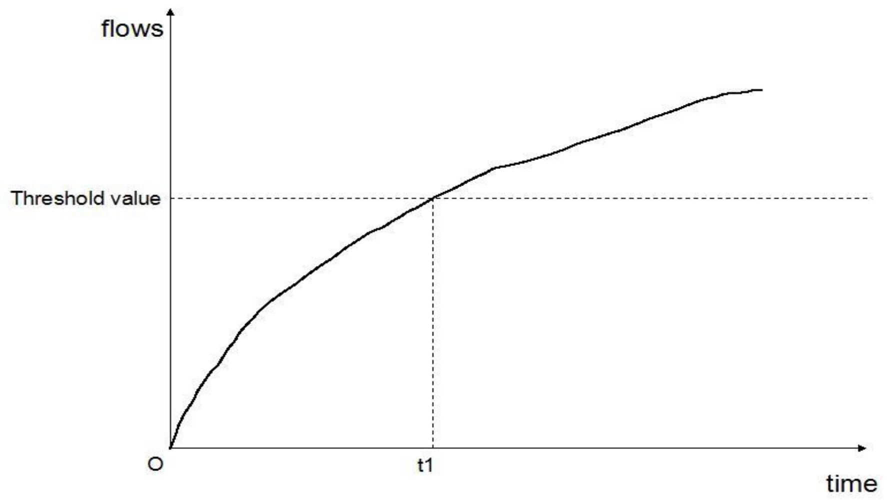

Figure 6. Inward Traffic-Flows Along with Time

Figure 6 shows that the inward traffic-flows increase as time goes on. As the controller does not realize that this switch's claim does not match its capacity, more and more flows are assigned to it regardless of its packet loss.

However, the outward traffic-flows of the same switch along with time are like Figure 7.

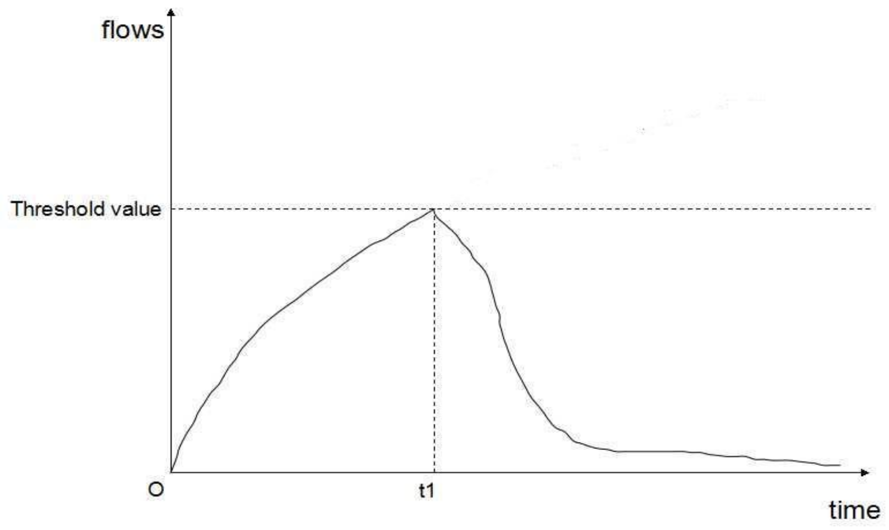

Figure 7. Outward Traffic-Flows Along with Time

The threshold value is the maximum bandwidth of this switch. Once the traffic-flows exceed the Threshold value, it is inevitable to lead to a large amount of packet loss. Thus, after time t1, the outward traffic-flows decrease sharply.

By implementing scheme 3.1, the controller will not assign flows through any switch beyond its threshold value. In this case, every switch can process all packets, which may reduce the impact by a malicious switch effectively. SDN with Threshold Value Control will perform an inward and outward traffic-flows trend curve like Figure 8.

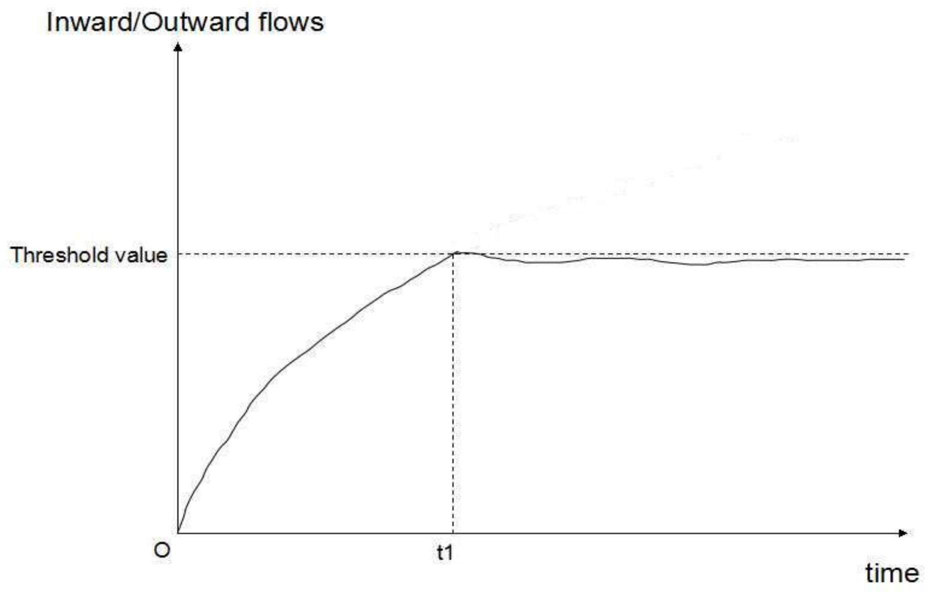

Figure 8. Inward and Outward Traffic-Flows of SDN with Threshold Value Control

By observing Figure 8, this scheme avoids packet loss by preventing the controller from assigning overload flows to a malicious switch. Comparing Figure 7 and Figure 8, we can conclude that the scheme in Section 3.1 acts as a malicious switch detector, and helps to improve the performance of SDN.

5. Related Work

The concept of SDN was firstly put forward by Nick McKeown, *et al.,* in [1]. [1] Not only stated the operating principle of Open Flow in details, but also enumerated the main application scenarios. It illustrated the promising future of an innovative network. Open Flow [2] becomes a milestone of SDN evolution. As Open Flow is still in its infant of development, there is much vulnerability. Kevin Benton [3] gave a detailed elaboration of these possible vulnerabilities. He is the first to focus on vulnerabilities that emerge from Open Flow network designs and vulnerabilities within the protocol [3]. Listed a lot of vulnerabilities like Man-in-the-middle Attacks, Listener Mode, Flow Table Verification, Denial of Service Risks, and Controller Vulnerabilities. Rowan Klöti1 made a security analysis on Open Flow in [4]. These papers were based on the view that all resources are limited including the sizes of the flow tables and bandwidths of the links.

Despite the extensive work in Open Flow with respect to monitoring, routing, and load-balancing, the intrusion detection field has not yet attracted much attention. A flow-based intrusion detection mechanism using Open Flow was reported in [5], which presented a lightweight method for DDoS attack detection based on traffic flow features with low overhead.

Suresh Kumar et al. also proposed an Open Flow switch with intrusion detection system in [6]. There are two new actions in the proposed switch, which are IP verification and Packet verification. And there are one more table other than flow table and an IDS database. The table is IDS IP information, which contains the list of suspicious IP. The IDS database contains the definition of various attacks. When a new packet arrives at the switch, the proposed Open Flow switch will check the source IP address in the IDS IP information Table. If the source IP is found in the table, which means that this packet is sent from a suspicious IP address, the switch will take drop packet action. If the source IP is not found in the IDS IP information Table, then it starts intrusion detection by checking the packet data with database. If intrusion is detected, then make an entry in IDS IP information Table and drop the packet. Otherwise, it processes the packet as normal processing. By adding this intrusion detection system, Open Flow switches become more secure and intelligent.

6. Conclusion

In this paper, we addressed the problem of detecting malicious switches in SDN as some malicious switches may be captured by an adversary and have a terrible influence on the network. We proposed two schemes to detect these suspicious switches and aimed to improve the communication quality of the network because the original SDN did not provide satisfied solution to this secure problem. After monitoring and investigating these suspicious switches, the controller takes further steps to verify and detect malicious switches.

For future research, we should put more attention on the security problems. We also plan to propose an effective method to handle malicious switches in SDN after they are detected by the schemes in this paper.

Acknowledgements

The research work reported in this paper is supported by National Science Foundation of China under Grant No. 61100172, No. 61272512, and No. 61300177. It is also supported by Program for New Century Excellent Talents in University (NCET-12-0046), and Beijing Natural Science Foundation (No. 4121001).

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, *et al.,* “Open Flow: enabling innovation in campus networks”, ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, **(2008),** pp. 69-74.
[2] “Open Flow Switch Specification”, http://www.openflowswitch.org/.
[3] K. Benton, L. J. Camp, C. Small, “Open flow vulnerability assessment”, Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM, **(2013),** pp. 151-152.
[4] R. Kloti, “Open Flow: A Security Analysis”, 8th Workshop on Secure Network Protocols (NPSec 2013), G?ttingen, Germany, **(2013)**.
[5] R. Braga, E. Mota and A. Passito, “Lightweight DDoS flooding attack detection using NOX/Open Flow”, Local Computer Networks (LCN), 2010 IEEE 35th Conference on IEEE, **(2010),** pp. 408-415.
[6] S. Kumar, T. Kumar, G. Singh, *et al.,* “Open Flow switch with intrusion detection system”, IJSRET, vol. 1, **(2012),** pp. 1-4.
[7] “Open vSwitch: An Open Virtual Switch”, [Online]. Available: http://openvswitch.org/.
[8] S. Das, G. Parulkar and N. Mckeown, "Simple Unified Control for Packet and Circuit Networks", IEEE Photonics Society Summer Topical on Future Global Networks, **(2009)** July.
[9] L. Chen and S. L. Ng, Comments on "Proving reliability of anonymous information in VANETs", by Kounga, *et al.*, IEEE Trans. Veh. Technol., vol. 59, no. 3, **(2010)** March, pp. 1503-1505.
[10] “A NICE Way to Test Open Flow Applications”, Available at http://code.google.com/p/nice-of/.
[11] “Floodlight-developers mailing list: Tls support”, **(2012)** January, https://groups.google.com/a/openflowhub.org/forum/.
[12] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, *et al.,* “Onix: A distributed control platform for large-scale production networks”, OSDI, **(2010)** October.
[13] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson and G. Gu. “A security enforcement kernel for openflow networks”, In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, New York, NY, USA, ACM, **(2012),** pp. 121-126.

[14] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown and G. Parulkar, "Flowvisor: A network virtualization layer. OpenFlow Switch Consortium, Tech. Rep, **(2009)**.
[15] B. S. Lee, R. Kanagavelu and K. M. M. Aung, "An efficient flow cache algorithm with improved fairness in Software-Defined Data Center Networks", Cloud Networking (CloudNet), 2013 IEEE 2nd International Conference on. IEEE, **(2013),** pp. 18-24.
[16] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma and S. Banerjee, "DevoFlow: scaling flow management for highperformance networks," SIGCOMM Comput. Commun. Rev., **(2011)** August.
[17] P. Dely, A. Kassler and N. Bayer, "Openflow for wireless mesh networks," in Proc. of ICCCN, vol. 31, **(2011)** August 4.
[18] K. Ramakrishnan, S. Floyd and D. Black, "The addition of explicit congestion notification (ECN) to IP," RFC 3168 (Proposed Standard), IETF, updated by RFCs, vol. 4301, no. 6040, **(2001)** September.
[19] J. Gettys and K. Nichols, "Bufferbloat: dark buffers in the internet," Communications of the ACM, **(2012)** January, vol. 55.
[20] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado and R. Sherwood, "On controller performance in software-defined networks," in Proc. of Hot ICE 2012, **(2012)**.
[21] R. Wang, D. Butnariu and J. Rexford, "Open Flow-based server load balancing gone wild", Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services. USENIX Association, **(2011),** pp. 12-12.
[22] D. M. F. Mattos, N. C. Fernandes, V. T. da Costa, *et al.,* "Omni: Open flow management infrastructure", Network of the Future (NOF), 2011 International Conference on the IEEE, **(2011),** pp. 52-56.
[23] M. Jarschel, F. Lehrieder, Z. Magyari, *et al.,* "A Flexible Open Flow-Controller Benchmark", Software Defined Networking (EWSDN), 2012 European Workshop on. IEEE, **(2012),** pp. 48-53.
[24] M. Canini, D. De Cicco, P. Kuznetsov, *et al.,* "STN: A Robust and Distributed SDN Control Plane", Open Networking Summit, **(2014)**.
[25] T. J. Kim, T. Lee, K. H. Kim, *et al.,* "An efficient packet processing protocol based on exchanging messages between switches and controller in Open Flow networks", Emerging Technologies for a Smarter World (CEWIT), 2013 10th International Conference and Expo on. IEEE, **(2013),** pp. 1-5.
[26] J. J. Halliday, S. K. Shrivastava and S. M. Wheater, "Flexible workflow management in the OPEN flow system", Enterprise Distributed Object Computing Conference, 2001. EDOC'01, Proceedings Fifth IEEE International IEEE, **(2001),** pp. 82-92.
[27] Z. Cai, A. L. Cox, T. S. E. N. Maestro, "A system for scalable Open Flow control", Technical Report TR10-08, Rice University, **(2010)**.

Authors

Xiaodong Du, he received B.S. degree in Computer Science and Technology from North China Electric Power University (China) in 2012. He is currently a M.S. student in School of Computer Science, Beijing Institute of Technology (China).

Mingzhong Wang, he received his Ph.D. degree in computer science from the University of Melbourne in 2010. Since 2010, he has joined School of Computer Science, Beijing Institute of Technology. His research interests include parallel and distributed processing, massive data processing, and information security.

Liehuang Zhu, he is currently a professor at School of Computer Science, Beijing Institute of Technology. He is an expert in network security. His research interests include security protocol analysis and design, group key exchange protocol, wireless sensor network and cloud computing.

Xiaoping Zhang, she is currently a graduate student at National Key Lab of Vehicular Transmission, China North Vehicular Research Institute. Her research interests include planet gear, vibration and simulation.