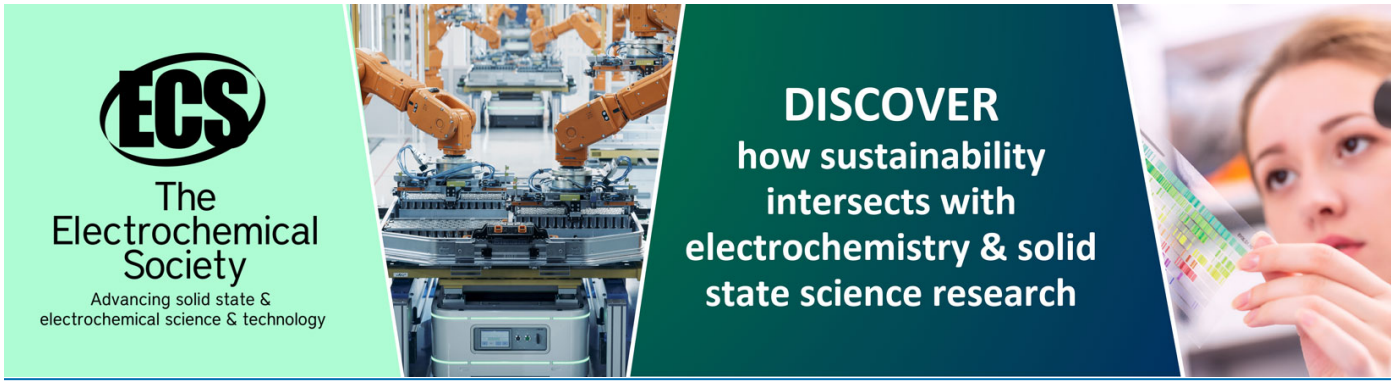
A Petri Net-based Method to Verify the Security of the SDN



A Petri Net-based Method to Verify the Security of the SDN

Ying Zhou[1,2,a], Yongqiang Bai[1,2,b] and Peng Zhao[3,c]

[1]Luoyang Electronic Equipment Test Center of China
[2]Luoyang Henan Province, China
[3]Beijing Jiaotong University, Beijing, China
[a]zy_jacksom0669@sina.com, [b]zy_jacksom0669@sina.com, [c]14125069@bjtu.edu.cn

Abstract. Due to the widespread research on Software Defined Networks (SDNs), the network security has been duly noted. But most of those efforts ponder over SDN security from the protocol perspective. To the best of our knowledge, none of the attempts has paid attention to the security analysis and modeling of state and communication in the SDN. Therefore, this paper provides a different approach to security analysis. Our objective is to analyze the security via the transformation state of communication model. Additionally, we present the security analysis method based on token to explore the potential threatens. Finally, we analyze the SDN via the combination of the number of token and time series based on Petri Net, and give the results. Our results are very bright in using such models to analyze such security objectives.

1. Introduction

The traditional IP network is complex and hard to manage. It is difficult to update because of the triple bindings, which are composed of the resource/location binding, user/network binding, and control/data binding [1]. To solve the complex problems, the proposals are discussed widely. Among of them, the separation of control plane and data plane is approbatory. Software Defined Networks (SDNs) is the representative, which brings growing concerns and higher expectations [2][3]. Recently, the security of the Internet has been paid more and more attention. The research on the security of the SDN occupies an important position.

OpenFlow is a typical protocol in the SDN. Several security updating approaches on OpenFlow have been provided. FRESCO was introduced in [4]. FRESCO is a security application development framework and provides a Click-inspired programming framework for security researchers to implement, share, and compose together. OpenFlow Random Host Mutation (OFRHM) was presented to defend against scanning-based attacks in [5]. Guang Yao et al. proposed VAVE [6] to improve the SAVI solutions by solving source address validation problems.

Methods to detect and defend against Distributed Denial of Service (DDoS) by the SDN have also appeared in recent years. A lightweight method based on traffic flow features was presented for detecting DDoS attacks in [7]. Suh et al. proposed a content-oriented networking architecture in [8], which reacted to resource exhaustive attacks like DDoS. Chu, YuHunag et al. proposed a research idea. It relies on OpenFlow and LISP technologies to implement a DDoS defender on an OpenFlow-enabled switch to realize an autonomic self-defense concept [9]. In other aspects, the authors in [10] performed a security analysis of OpenFlow by using STRIDE and Attack tree modeling.

However and to the best of our knowledge, there is still little attention given to the security analysis and modeling of state and communication in the SDN. Therefore, we present a security analysis model based on Petri Net [11]. First, we abstract the SDN into a simple network structure to get a basic communication structure. Then, we build a communication model and detailed three running states. Finally, we analyze the security of the SDN and derivate the potential attacks on the basis of the model.

The rest of the paper is given as follows: we present description and analysis of our method in Section II. Section III analyzes the security of the SDN based on the method. Finally, we conclude this paper in Section IV.

2. Description and Analysis of the Method

This section will show the details of the analysis method. There are two parts, which is simplifying the network structure and Modeling the Petri Net for the SDN.

2.1. Simplifying the Network Structure

The SDN decouples the control plane and data plane. Based on the both planes, we divide the network topology into two parts. The control plane includes Controller and Switch. Switch and Terminal are assigned to the data plane. The topology is shown in Fig.1. The Switch is the bridge between Controller and Terminal and it belongs to the both planes. Data flow is transferred in two-ways between Terminal1 and Terminal2 by the Switches. Command flow exists between Controller and Switches.

Simplification of network structure for the SDN is the basis to know the communication process. It is very useful for the researchers to analyze security for the communication process.

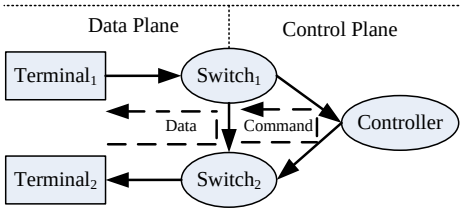


Figure 1. Logic structure diagram of the SDN

2.2. Modeling Petri Net for the SDN

Based on the logic structure of the SDN in section A, modeling the communication of the SDN by Petri Net will be shown in this section. The analysis by Petri Net includes the graphics, logic, and so on. These are suitable for analyzing the synchronous, asynchronous, distributed and concurrent computer systems [12]. The characteristics in above are predominant to show the communication state of different data types and the logical relationship between each other in the communication process of the SDN.

2.2.1.The Communication of the SDN. The communication of the SDN can be divided into six parts. We assume that data is sent from Terminal1 to Terminal2 in Fig.1.

1-After receiving the messages from Terminal1, Switch1 checks whether there is a flow entry for Terminal1 and Terminal2 in the flow table. If not, the access request will be sent to Controller to query the data transmission path for Terminal1 and Terminal2 by Switch1.

2-After receiving the request, Controller will check on the corresponding flow entries to send them to Switch1.

3-Switch1 will update the local flow table, until receiving the response message from Controller. The data from Terminal1 will be transmitted to ARN2.

4-The same as the 2nd step, Switch2 will check the local flow table to look for the path entries. If not, Switch2 will send the request to Controller.

5-Controller sends the command to Switch2.

6-After getting the command, Switch2 will revise the local table like Switch1 in the 3rd Step. The messages are forwarded to Terminal2 and the communication ends.

The path messages will be stored in the flow tables until the end of the cache timer. During this time, Switch1 and Switch2 need not send the request messages to Controller again.

2.2.2. Petri Net Model of the Communication. As the steps listed in the previous sub-section, we build a Petri Net model of the communication in Fig.2. To meet the model requirement in the SDN, we define the conditions of Transitions and the functions of Places, and adjust the weight of Arcs in the Petri Net model of the communication.

∑=(P,T;F,K,W,M0) is used to stand for this model. The definitions on the whole elements in ∑=(P,T;F,K,W,M0) are as follows.

P is a finite set of Places;

T is a finite set of Transitions;

F is a set of Arcs;

N=(P,T;F) is a net, which is the underlying net of ∑=(P,T;F,K,W,M0);

K is the capacity function of N;

W is the weight function of N;

M0 is a finite set of the initial tokens, which satisfies K.

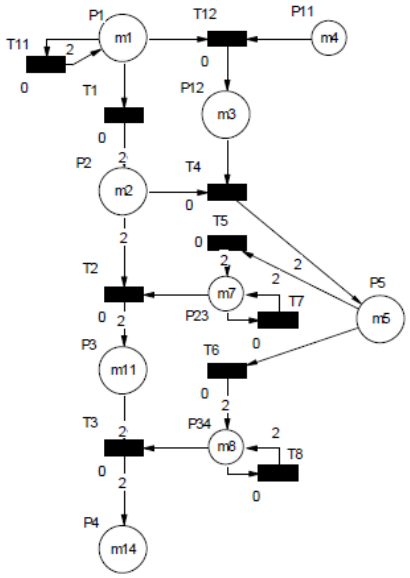


Figure 2. Petri Net model of the communication

Table 1. Description of places

Place	Description
P1	The input places which send data tokens
P2/P3	The places holding the forwarding data tokens
P4	The output place which receives data tokens
P5	The place holding the authentication and path tokens
P11	The input places which send authentication request tokens
P12	The place holding the authentication request token
P23/P34	The places holding the path tokens

Table I and II give the detail description of places and transitions respectively. The default value of K is ∞, where K(P1)=K(P4)=K1, K(P2)=K(P3)=K2, K(P5)=K3, K(P23)=K(P34)=K4, K(P11)=K(P12)=K5. The default value of W is 1, where W(T11,P1)=W(T1,P2)=W(P2,T2)=W(T2,P3)=W(P3,T3)=W(T3,P4)=W(T4,P5)=W(T5,P23)=W(T7,P23)=W(T6,P34)=W(T8,P34)=2. Limited by the simulation environment, the type of W has to be set as integer, such as 1, 2, et al. The default value of M0 is 0, where M0(P1)= 3, M0(P11)=1.

Table 2. Description of Transitions

Transition	Description
T1	Sending the data token to P2
T2	Sending the data token to P3
T3	Sending the data token to P4
T4	Sending access request to P5
T5	Sending the path token to P23
T6	Sending the path token to P34
T7/T8	Sending the path token to P23/P34
T12	Sending the authentication request and access tokens
T11	Sending the data token to P1

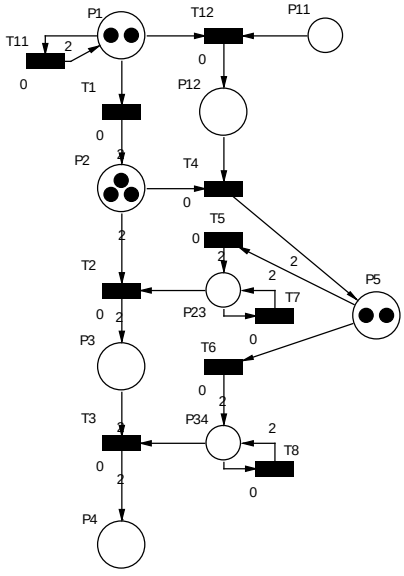


Figure 3. The access state

2.2.3. *Running the Petri Net Model.*Fig.2 is the initial state. Once running, P1 sends a data token to P2 by T1 and P11 sends a data token to P12 with P1 by T12. Then P12 and P2 send an access request token to P5. Petri Net model is in the access state in Fig.3.

In the access state, T5 and T6 have reached the transition conditions and send path tokens to the next Place. The model enters the activation state in Fig.4. In the access state, although the data token has been in P2, P2 is unable to forward it without the path rules. It will be able in the activation state. After receiving the path tokens, P2 will send the data tokens to P3 through T2, and P3 forward them to P4 through T3. The data tokens arrive at the destination Place. Because the access and path tokens

have been produced and stored. When the following up data tokens are sent by P1, the access and activation states will be skipped, and Petri Net model will get into the stable state shown as Fig.7.

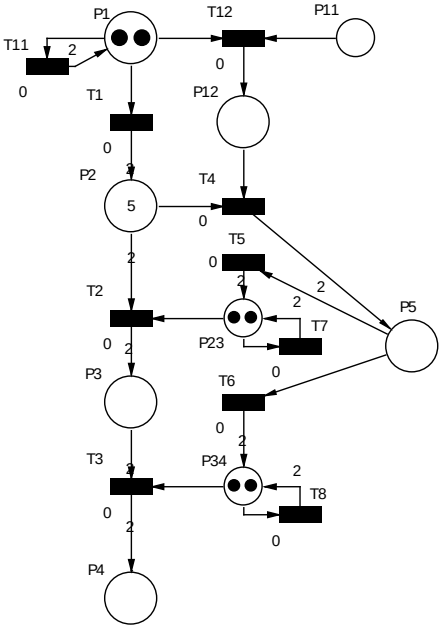


Figure 4. The activation state

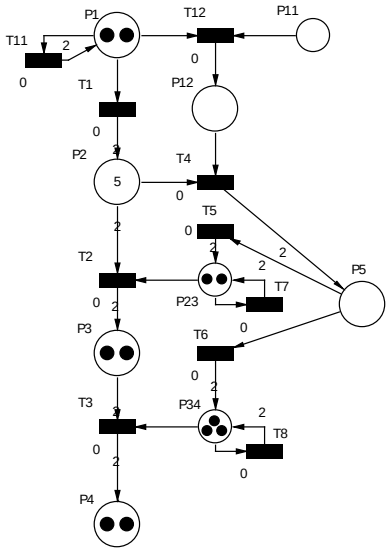


Figure 5. The stable state

3. Analyzing the Security Based on the Number of Tokens

From Fig.2-Fig.5, we can see that if we add the number of inputting Place such as P1, the access and path tokens in P23, P34 and P5 will increase. The number of Transitions also increases. However, a capacity limits Place in the Petri Net model. For example, K(P5)=K3 in our models. On the other way, we can find that the increasing number of combination in Fig.5 including P1, P11 and T12, will also

occupy the token capacity of P5. When the limit is exceeded, the model will not work normally. The token capacity in Petri Net is similar to the resource in the network device. The number of the combination is to the number of the terminal. And the capacity of P5 is to the resource of Controller in the SDN. The network will be dangerous when it is attacked by DDoS, Flooding and so on.

From Fig.2 to Fig.5, the running state needs the time. The process to wait for the access and path tokens consumes several steps. The steps could be considered as the network delay. The attackers could make use of the delay to forgery the information and threaten the network security.

4. Conclusion

In this paper, we presented a Petri Net-based network security analysis method for the SDN. First, we abstracted the SDN into a simple network structure with two terminals, two Switches and one Controller to get a basic communication structure. Then, we built a Petri Net model according to the communication structure, and detailed three running states. Finally, we analyzed the security of the SDN and derivated the potential attacks by the numbers and time series of token.

The next research includes repeating the potential attacks as we proposed in section II in the simulation and test bench to analyze the influence. It would be helpful for the researcher to improve the security of the SDN.

5. References

[1] Zhang, Hongke, et al. “Smart identifier network: A collaborative architecture for the future internet.” IEEE Network 30.3 (2016): 46-51.

[2] Guck, Jochen W., et al. “Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation.”IEEE Communications Surveys & Tutorials 20.1 (2018): 388-415.

[3] Amin, Rashid, Martin Reisslein, and Nadir Shah. “Hybrid SDN Networks: A Survey of Existing Approaches.” IEEE Communications Surveys & Tutorials (2018).

[4] S. Shin, P. Porras, V. Yegneswaran, et al., “FRESCO: Modular Composable Security Services for Software-Defined Networks,” NDSS. 2013.

[5] J.H. Jafarian, F. Al-Shaer, Q. Duan, “Openflow random host mutation: transparent moving target defense using software defined networking,” Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012: 127-132.

[6] G. Yao, J. Bi, P. Xiao, “Source address validation solution with OpenFlow/NOX architecture,” Network Protocols (ICNP), 2011 19th IEEE International Conference on. IEEE, 2011: 7-12.

[7] R. Braga, E. Mota, A. Passito, “Lightweight DDoS flooding attack detection using NOX/OpenFlow,” Local Computer Networks (LCN), 2010 IEEE 35th Conference on. IEEE, 2010: 408-415.

[8] Y. Choi, “Implementation of Content-oriented Networking Architecture (CONA): A Focus on DDoS Countermeasure,” Proceedings of European NetFPGA developers workshop. 2010.

[9] Y.H. Chu, M.C. Tseng, Y.T. Chen, Y.C. Chou, Y.R. Chen, “A novel design for future on-demand service and security,” 2010 IEEE 12th International Conference on Communication Technology. 2010: 385-388.

[10] P. Porras, S. Shin, V. Yegneswaran, et al., “A security enforcement kernel for OpenFlow networks,” Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012: 121-126.

[11] Narayanan, Murale, and Aswani Kumar Cherukuri. “Verification of Cloud Based Information Integration Architecture using Colored Petri Nets.” International Journal of Computer Network and Information Security 10.2 (2018): 1.

[12] Yao, Linyuan, et al. “Security Analysis Based on Petri Net for Separation Mechanisms in Smart Identifier Network.” Computer Communication and Networks (ICCCN), 2017 26th International Conference on. IEEE, 2017.