SDN Dynamic Access Control Scheme Based on Prediction

Qian Cui, Shihui Zheng(✉), Bin Sun, and Yongmei Cai

[1] School of Computer Science and Engineering,
Xinjiang University of Finance and Economics, Urumqi 830000, China
shihuizh@bupt.edu.cn

Abstract. Through research on the access control of software defined network (SDN) northbound interfaces, we found that malicious OpenFlow applications (OF applications) abuse the northbound interfaces with ADD permissions, which can cause the controllers function failure and other serious harm or even crash directly. Most previous studies of this issue, such as those resulting in the ControllerDAC scheme, set static thresholds; and did not find effective solutions to those problems. This paper analyzes the characteristics of the input flows and proposes an SDN dynamic access control scheme based on prediction and dynamic adjustment of the load threshold. By examining the access characteristics of the OF application, we use a prediction algorithm to determine whether the application will disrupt the API with ADD permissions. This algorithm enables us to perform targeted dynamic access control for different types of applications. Experimental results show that compared with the aforementioned ControllerDAC scheme, our scheme effectively reduces the malicious flow table rate and limits the delivery of malicious flow tables, and the extra delay generated by our scheme is less than 10%.

Keywords: Software defined networking · OpenFlow application · Flow entry prediction · Network security

1 Introduction

Software defined networking (SDN) is an emerging software-based architecture and technology. SDN's most notable feature is that the control plane and data plane support a loosely coupled, centralized network state control, resulting in a transparent the upper-layer network infrastructure. An OF application can freely add and modify the flow table in the switches through the controller's open northbound interface. We find that the number of flow tables in the different controllers switch connections affects the controller's function; and what's worse it may cause the controller to crash. In paper [4] OpenvSwitch and Floodlight Controllers can store 140,000 flow rules, but once the malicious OF application inserts the flow table exceeds this limit, the new flow table will replace the original flow table or even the firewall flow table, thus destroying the security rules of the network itself. In our test when the RYU controller was connected to the OVS switch, that has more than 60,000 flow tables, the controller automatically deletes the previously stored flow tables damaging the network structure. When an

S. Liu and G. Yang (Eds.): ADHIP 2018, LNICST 279, pp. 57–67, 2019.
https://doi.org/10.1007/978-3-030-19086-6_7

ONOS controller [9] has more than 45,000 entries in the switch, there will be serious consequences of a direct controller crash. Most physical OF switches currently on the market typically have small flow table storages of approximately 8000 entries [12]. Protecting the controller from the malicious behavior of an OF application is a problem that needs to be solved.

[1–3, 6] the extension of security functions for controllers, such as Floodlight and NOX, enhance controller security by modifying controller's source code related functions. This method is affected by the controller's system architecture, programming language, etc. Building extensions is difficult and will increase the load on the controller. The idea [7, 8] of decoupling the security plane from the control plane is proposed. Such as controllerSEPA scheme, filter OF applications using security schemes. Repackaged and delivered to the controller through a secure northbound interface. This type of security scheme increases deployment flexibility and facilitates porting to any controller. Possible security issues with access controllers by analyzing OF applications, provides protection such as authentication, role-based authorization, etc. However, the paper [4] points out the defects in the above access control methods. In other words, the features of the interface that does not incorporate the controller provides more targeted protection. For example, an application with delete permission can still delete a flow table with a higher priority than him. Based on the functions of the northbound interface provided by the controller, this paper classifies the OF application access controller resources into four categories, that is, READ, ADD, UPDATE and REMOVE, and designs a corresponding security plan for each type of threat. This scheme uses a policy engine and static thresholds to set a more granular access policy for OF applications, effectively protecting the controller. However, static thresholds have limited protection for ADD permissions. For example, the ControllerDAC scheme sets the threshold for sending 50 flow rules in 60 s for DEFAULT classes of OF applications. For ONOS controller who only need twelve hours to make the flow OpenVswitch elevate to the upper limits. Therefore, when an attacker controls multiple applications, it requires only several hours to reach the switch's storage limit and cause the controller to crash. Table 1 summarizes the four APIs that we need to protect against abuse of ADD permissions.

Table 1. Controller needs to protect ADD permissions.

Controller	ADD permissions that need protection
OpenDaylight	/sal-flow:add-flow
ONOS	/flows/<deviceId> /link
Floodlight	/wm/staticflowpusher/json
RYU	/stats/flowentry/add /stats/groupentry/add

Our paper analyzes the input flows characteristics of OF applications [5] in relation to ADD permission problems. By using the Moving Average Algorithm, wen can

predict the number of flow tables in time series. Determine whether the OF application is a malicious application, then we use the predicted number of flow tables as a basis to dynamically adjust the threshold. We decouple the security plane from the control plane, combining policy templates and dynamic thresholds to control the operation of the OF application's ADD access to the controller. Make the number of flow tables in the switch safe and reasonable. Our solution separates OF applications from controllers, did not modify the source code of the controller. Therefore, our solution is applicable to any controller.

This article is structured as follows: The first section describes the main problems existing in OF applications and our solutions. In section two, we propose a design and describe its prototype implementation. The third section simulates the abuse of ADD permissions by malicious OF applications, and make comparisons with the Controller DAC solution, and RYU controller solution to verify our proposed solution. The last section presents the study's conclusions.

2 SDN Dynamic Access Control Scheme Based on Prediction

The SDN dynamic access control scheme proposed by our paper, is composed of a dynamic access control module and a flow table threshold prediction algorithm module. Figure 1 depicts the scheme's architecture. Flow table threshold prediction algorithm modules use the number of flow tables delivered in each cycle, to estimate the number of arrivals for the next periodic flow table. After estimating the number of flow tables, an abnormal, determination causes an adjustment to the threshold of the OF applications. Dynamic access control module provides OF applications authentication and dynamic access control. If the OF application access count is less than the threshold, it allows the OF application normal access to the controller. If the number of accesses is greater than the threshold, the application is prevented from using the ADD permission to send the flow table, protect the controller from the abuse of ADD permissions.

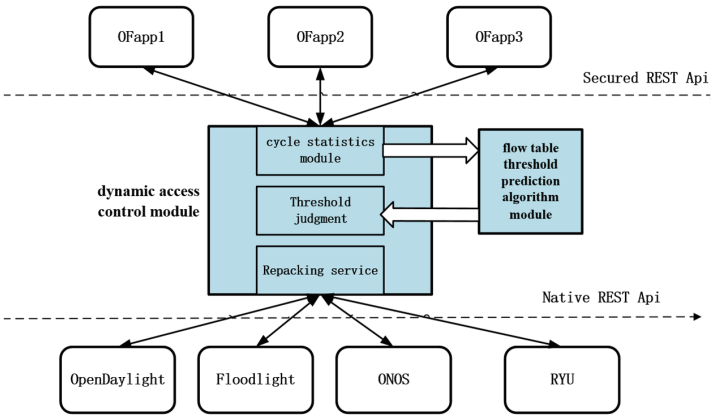

Fig. 1. SDN dynamic access control architecture based on prediction

2.1 Flow Table Threshold Prediction Algorithm

Flow table threshold prediction algorithms estimate the number of flow tables for the next period of the OF application through the Double Moving Average algorithm. The threshold is subsequently adjusted by a combination of trend changes and safety factors.

Data Analysis Based on Double Moving Average Algorithm
The flow table threshold prediction algorithm first enters the number of flow tables Y_t and spanning cycles n applied by the OF application through the ADD permission during the input period. Next, the Double Moving Average algorithm is used to analyze the historical data and, to outputs the predicted value at the next moment, the intercept a_t and the trend of change b_t. The specific algorithm process is as follows:

Let Y_t be the value of the flow table in the t th period, and the moving average $M_{t+1}^{(1)}$ of the $t+1$ period is:

$$M_{t+1}^{(1)} = \frac{Y_t + Y_{t+1} + \ldots + Y_{t-n+1}}{n} \tag{1}$$

n is the span of the moving average, and the larger n is, the better the smoothing effect is. To avoid systematic errors, a further moving average is based on the first average:

$$M_{t+1}^{(2)} = \frac{M_t^{(1)} + M_{t-1}^{(1)} + \ldots + M_{t-n+1}^{(1)}}{n} \tag{2}$$

Contained in the formula, $M_{t+1}^{(2)}$ is the double moving average of t th period, secondary motion correction eliminates the lag predictive value. The prediction model of the double moving average algorithm is:

$$M_{t+T} = a_t + b_t \cdot T \tag{3}$$

$$a_t = 2M_{t+1}^{(1)} - M_{t+1}^{(2)} \tag{4}$$

$$b_t = \frac{2}{n-1}\left(M_{t+1}^{(1)} - M_{t+1}^{(2)}\right) \tag{5}$$

T is the mean to move backwards, $M_{t+1}^{(1)}$ is the last average of the series of calculated moving averages. $M_{t+1}^{(2)}$ means the last double moving average in the calculated sequence of the double moving average. a_t is the intercept, used to modify the initial point of prediction to eliminate the hysteresis of the predicted value. b_t is the trend change.

It should be noted that during the periodic sampling of the outgoing flow table inserted by OF application, if there is no sample before the sampling period, and historical data is insufficient, the front flow table of the number of cycles is set to zero. Through the Double Moving Average algorithm, when $T = 1$, we obtained the forecast value of the flow tables for the next OF application.

Analysis of Threshold Dynamic Adjustment

Next, we enter the time predicted value M_{t+1}, the trend of change b_t and safety factor α to analyze the ADD operation of the application, and limit the operation of the OF application through the output threshold S_{t+1}, Specific analysis is as follows:

When the predictive value of the OF application is greater than the initial threshold set by the system, we can determine that the OF application abuses the ADD permission. To ensure the safety of the controller, it is necessary to reduce the threshold appropriately to militate against the threat generated by malicious OF applications.

This article proposes the concept of safety factor α. The fact adjusts according to the level of safety required in the controller's environment. Change trend b_t is used to indicate the degree about abuse of ADD permissions or abuse of OF applications, these factors act in unsion to adjust the threshold.

$$\begin{cases} S_{t+1} = S_t - (\alpha + |b_t|)\, M_{t+1} > S_{thr} \\ S_{t+1} = S_t + (\alpha + |b_t|)\, M_{t+1} < S_{thr} \end{cases} \tag{6}$$

S_t is the threshold at time t, S_{thr} is the initial threshold, M_t is the predicted value of flow table at time t, and $S_{thr} > S_t > 0$.

When the abuse of the control plane occurs, the flow table threshold prediction algorithm actively reduces the threshold to limit the delivery of malicious flow tables. When the abuse operation is released, the threshold prediction algorithm will actively increase the threshold (but will not exceed the application's default threshold) to ensure that the OF application normally accesses the controller.

2.2 Dynamic Access Control Module

The dynamic access control scheme filters the OF application using the authentication and authorization scheme. Combined with the threshold predicted by the threshold prediction algorithm, this limits the ADD permissions applied by the OF. The scheme consists of an OF application filtering module and periodic statistics module.

The OF application filter module applies an application name to each OF, and starts a thread to authenticate and authorize the accessed OF application.

We use the username and password to verify the validity of the OF application. Next, we use the role to distinguish between different OF applications. Every service request of the OF application, we requires verification of the application related information, such as: application name, permission, and access threshold. We refer to paper [10] to propose two roles:

ADMIN: The administrator authority has the highest security level. By default, the OF application with this authority can freely perform the ADD operation on the controller through the security extension without constraints.

DEFAULT: This authority is set to access for general applications. OF application of this type of role Prone to abuse ADD permission. In view of this situation, we will set a default threshold for this type of authority, and the default threshold will be changed according to the degree of abuse of the OF application.

Figure 2 shows the template for different OF application permissions and initial threshold settings. The template defines the application name OFapp2 of the OF

application, the application role DEFAULT and the initial threshold set for this role, that is, this initial threshold settings allows OFapp2 to use ADD permissions 50 times every 60 s.

```
"apps":[
{
"appname":"ofapp1",
"api-roles":"admin",
"threshold seconds":"",
"times":"",
},
{
"appname":"ofapp2",
"api-roles":"default",
"threshold seconds":"60",
"times":"50",
}
]
```

Fig. 2. OF application access policy template

The OF application filter module first parses and counts OF application names, this module will only parse requests related to ADD operations, and we count the number of periodic visits of the OF application through the OF application name.

Periodic statistics modules will additionally enable a thread, This module will clear the record of the number of visits per period OF application, in order to ensure that the number of flow tables recorded in the next cycle will not be affected by the previous cycles. The application flow chart is as follows (see Fig. 3).

The overall scheme is described below: First, we filter all OF applications, OF application filtering modules will authenticate every request from OF app as well as checking the request's legality, and will record the total number of OF application accesses in the recording period through thread one. Next, we calculate the adjustment threshold through the flow table threshold prediction algorithm, then we clear the number of flow records recorded in the previous cycle to record the number of accesses for the next cycle. Adjust the threshold for the next cycle.

The second thread is the primary thread for dynamic access control. This thread will first record the number of accesses for each OF application, by judging with the thread's calculated threshold one. If the number of accesses is less than the threshold, it is allowed to go to the controller, otherwise it will prevent the delivery.

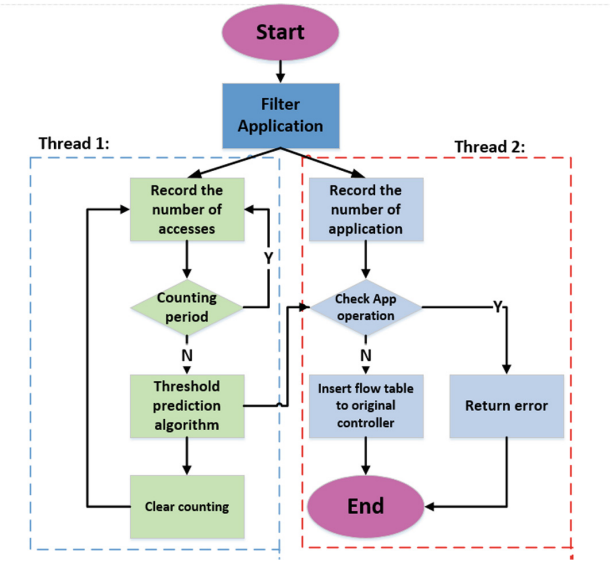

Fig. 3. Dynamic access control flow chart

3 Prototype and Experimental Validation

3.1 Experimental Environment and Parameters

To verify our solution, we first select the RYU controller, the switch selected is OpenvSwitch V2.0.2 used to store our repackaged ADD request. We chose the RYU controller and OVS switch as a direct connect solution.

This paper examines three different types of OF applications, shown in Table 2. We use ADD permissions to insert flow tables. Normal OF applications do not insert excessively many flow tables [11] within a certain period. Therefore, the methodology of constructing a flow table is: The normal application of the property we set the number of insert stream table is less than the default threshold, the flow table inserted by the abnormal applications will far exceed the default threshold.

Table 2. Three application under the flow table

OF application	Application properties	Authority	Threshold	Insert flow table
OFapp1	Normal	ADMIN	Null	20/cycle
OFapp2	Malicious	DEFAULT	50	Linear growth
OFapp3	Normal	DEFAULT	50	20/cycle

OFapp1 is the normal application with ADMIN authority. And use ADD permissions to insert the flow table at a frequency of 20 times/cycle. We do not set a threshold for this type of application. OFapp2 and OFapp3 have DEFAULT permissions.

OFapp2 is a normal application, the frequency of using ADD permission is 20 times/cycle. OFapp3 is a malicious application, and the number of inserted flow tables increases linearly with time.

3.2 Performance

This paper compares the proposed scheme with the static threshold scheme of the ControllerDAC and the non-threshold RYU controller by using three indicators: the delay of the security scheme, the number of flow tables stored by the switch and the malicious flow table rate.

Delay of the Security Scheme
We insert flow tables 10 times through the OF application and record the time delay of each access.

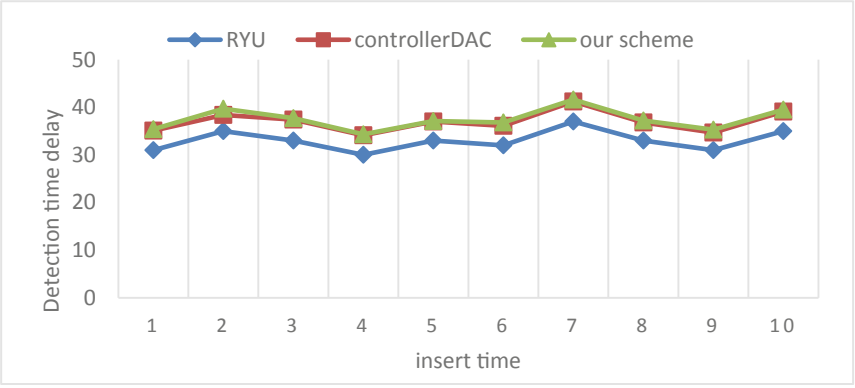

Fig. 4. Comparison of delays of the three schemes

It can be shown in Fig. 4, our scheme is compared to the ControllerDAC scheme, each access delay is within the range 1 ms. Compared to the original RYU controllers, our delay averages approximately 3–4 ms, and the average delay is approximately 10%, the delay is very small, and within the acceptable range. This property is observed because the main algorithm is calculated in another thread. The main thread only needs to obtain the threshold of the flow prediction algorithm to filter the application. Therefore, the impact of our scheme on the access efficiency of the OF applications is negligible.

Number of Flow Tables Stored by the Switch
We used the OF application of Table 2 to perform ADD access operations on the three schemes. We analyzed the number of flow tables existing in the switch, to verify whether our scheme can guarantee the flow of the switch within a safe range. The experimental results are shown in Fig. 5.

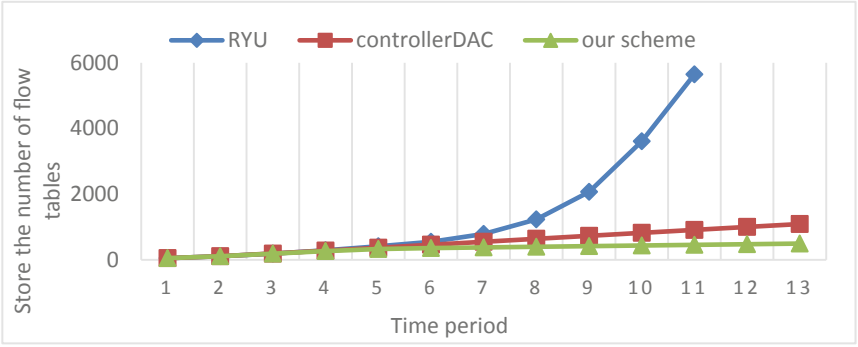

Fig. 5. Comparison of the number of flow entries stored in a switch

From the sixth observation period, our flow table threshold prediction algorithm detected an abnormal access by a malicious application. The number of RYU controller flow tables without any security protection after the next five cycles reached 7,000 and soon reached the limit of the flow table that the controller could tolerate. The DAC scheme has about 1000 flow lists after five cycles, compared with the RYU controller. The number of switches in the flow table is reduced, but it is still grows linearly, therefore over time, it reaches reached the upper limit of the controller. However, in the five cycles of the OF application abuse, the number of flow tables in our scheme has always kept a small amount, and tends to be stable. Therefore, our scheme was the best in protecting the controller when there is a malicious flow table.

The main reason is that in a source RYU controller without any protection, the controller is open to any OF application, so the malicious OF application can freely add the flow table without any control. In the ControllerDAC scheme, the setting for ADD permissions is static, this scheme does not identify malicious applications. All applications are issued according to the limit of 50 flow tables per period. Therefore, malicious flow tables will continue to increase over time and threaten the functionality of the controller. In our scheme, we detect a malicious OF application through the flow table threshold prediction algorithm. Dynamically reducing the threshold to zero prevents malicious flow table access, and effectively controls the number of flow tables in the switch, preventing controller crashes and other issues.

Malicious Flow Table Rate

The malicious flow table rate refers to the ratio of the number of flow tables generated by malicious applications to the total number of switch flows in the flow table stored in the switch. In general, malicious applications use switch resources and affect switch performance. Figure 6 compares the malicious flow table rates.

The formula for calculating the malicious flow rate is as follows:

$$FMFR = D_{af}/D_f \tag{7}$$

FMFR is the flow table match rate, D_{af} is the number of malicious flow tables in each cycle, and D_f is the total number of flows issued by all flow tables.

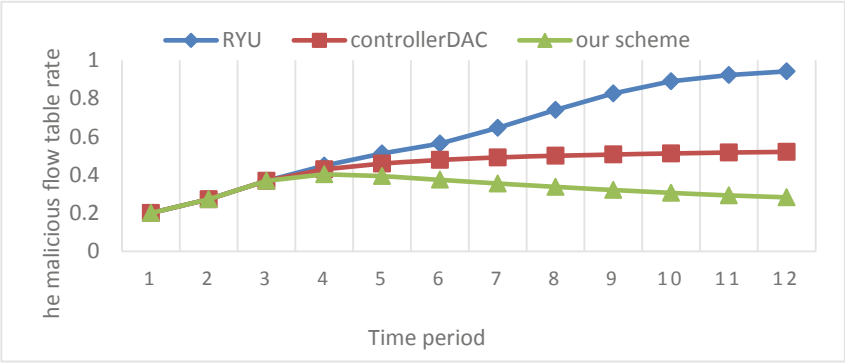

Fig. 6. Malicious flow table rate

It can be seen from Fig. 6 that the rate of malicious flow tables in the RYU controller remains at a notablely high level over time, and at the highest level, the percentage of malicious flow tables is close to 90%. The proportion of malicious flow tables for ControllerDAC reaches 40% to 50%, while the malicious flow table rate for our scheme remains below 40%.

In the RYU scheme, there is no restriction on Ofapp2, and the ControllerDAC scheme cannot distinguish the abnormal OF applications. Therefore, these abnormal flow tables enter the switch through the controller, occupy the switch resources, and threaten the security of the controller. In our scheme, the threshold is adjusted immediately after the abnormality of Ofapp2 is found and prevents insertion of malicious flow tables from being insert. At the same time for normal applications of ADMIN permissions OFapp1 we allow normal insertion of flow rules. Therefore, the malicious flow rate in the switch is effectively reduced.

4 Conclusions

This paper studies the northbound interface and determined that OF applications endanger the controller and network security by abuse ADD permissions. We designed a set of prediction-based dynamic access control schemes through analysis of security issues and comparing previous studies. The experimental results show that our solution evaluates the flow tables of malicious OF applications in a timely manner and implements targeted control based on the security authority. On the basis of guaranteeing efficiency, we can reduce the delivery of malicious flow tables. At the same time, our scheme separates the application layer from the control layer and can be flexibly deployed on most controllers. In the future, we plan to design a security plan for READ, UPDATE, and REMOVE permissions and develop a complete northbound interface security control scheme.

References

1. Scott-Hayward, S., Kane, C., Sezer, S.: Operationcheckpoint: SDN application control. In: Proceedings of the 2014 IEEE 22nd International Conference on Network Protocols, Ser., ICNP 2014, pp. 618–623 (2014)
2. Noh, J., Lee, S., Park, J., et al.: Vulnerabilities of network os and mitigation with state-based permission system. Secur. Commun. Netw. **9**(13): 1971–1982 (2016)
3. Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M., Gu, G.: A security enforcement kernel for OpenFlow networks. In: Proceedings of the 1st Workshop on Hot Topics in Software Defined Networks, pp. 121–126. ACM, Helsinki (2012)
4. Tseng, Y., Pattaranantakul, M., He, R., Zhang, Z., Nait-Abdesselam, F.: Controller DAC: securing SDN controller with dynamic access control. In: IEEE ICC 2017 Communication and Information System Security Symposium (2107)
5. Alfred, R., Fun, T.S., Tahir, A., et al.: Concepts labeling of document clusters using a hierarchical agglomerative clustering (HAC) technique. In: Uden, L., Wang, L., Corchado Rodríguez, J., Yang, H.C., Ting, I.H. (eds.) The 8th International Conference on Knowledge Management in Organizations, pp. 263–272. Springer, Dordrecht (2013). https://doi.org/10.1007/978-94-007-7287-8_21
6. Porras, P., Cheung, S., Fong, M., Skinner, K., Yegneswaran, V.: Securing the software-defined network control layer. In: Proceedings of the 2105 Annual Network and Distributed System Security Symposium (NDSS 2015), pp. 1–15. Internet Society, San Diego (2015)
7. Banse, C., Rangarajan, S.: A secure northbound interface for SDN applications. In: Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA (2015)
8. Tseng, Y., Zhang, Z., Nait-Abdesselam, F.: ControllerSEPA: a security-enhancing SDN controller plug-in for OpenFlow application. In: Proceedings of the 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (2016)
9. ON.Lab: ONOS application permissions. https://wiki.onosproject.org/display/ONOS/ONOS+Application+Permissions
10. Porras, P., Cheung, S., Fong, M., Skinner, K.: Securing the software-defined network control layer. In: Proceedings of the 2015 Network and Distributed System Security Symposium (NDSS), February 2015
11. Benson, T., Akella, A., Maltz, D.A.: Network traffic characteristics of data centers in the wild. In: Proceeding of the 10th ACM SIGCOMM Conference on Internet Measurement, pp. 267–280. ACM (2010)
12. Kreutz, D., Ramos, F.M.V., Veríssimo, P., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. CoRR, vol. abs/1406.0440 (2014). http://arxiv.org/abs/1406.0440