Toward a Trust-Based Authentication Framework of Northbound Interface in Software Defined Networking

Phan The Duy(✉), Do Thi Thu Hien, Nguyen Van Vuong, Nguyen Ngoc Hai Au, and Van-Hau Pham

Information Security Laboratory, University of Information Technology, VNU-HCM, Ho Chi Minh City, Vietnam
{duypt, hiendtt, haupv}@uit.edu.vn,
{14521108, 14520041}@gm.uit.edu.vn

Abstract. Software Defined Networking (SDN) – a new rising terminology of network is recently gained more and more interest in both academic and industrial field. Not only decoupling of its control plane and data plane, SDN also provides the whole view of entire network for better and more flexible network management. Despite the benefits of the global view of the whole network, SDN with a single point of failure at the controller encounters some drawbacks and additional challenge for security. A malicious OpenFlow application (OF app) can access to SDN controller to perform illegal activities due to the lack of the authentication protocol in Northbound interface to ensure that only trusted, and authorized applications access critical network resources. The information about the whole network, such as topology data, flow information or statistics can be retrieved. Even worse the entire network can be controlled from the compromised controller. In this paper, we introduce Trust Trident - a framework of securing trustworthy authentication between applications and controller, with the controller-independent capability. It gives network administrator a fully and fine-grained observation of OF apps communicating with the controller. Threats in Northbound interface and counter measurements by our plugin are classified and evaluated according to the threat categories from the STRIDE methodology.

Keywords: Northbound interface · Trust authentication · SDN

1 Introduction

Conventional network architecture has remained mostly unchanged over the last decades from its launching stage and proved to be difficult to monitor in large scale size network. In this circumstance, SDN has recently emerged to become one of the promising technologies for the 5G era and the future Internet. The architecture of SDN, shown in Fig. 1, separates the network control and forwarding functions from network device, which allows the network to be programmed by the application and monitored from a centralized controller. However, like any other centralized architecture, the advent of controller also brings up issues of security and availability for the network.

T. Q. Duong et al. (Eds.): INISCOM 2019, LNICST 293, pp. 269–282, 2019.
https://doi.org/10.1007/978-3-030-30149-1_22

According to Kreutz et al. [1], SDN itself may be a target of some threat attack vectors. Controller, controller-data interface and controller-application interface are identified as three of critical positions which can be easy to exploit. In the survey of SDN security [2], Scott-Hayward et al. described a categorization of the potential attacks to which the architecture is vulnerable. It is composed of unauthorized access, data leakage, data modification, malicious/compromised applications, denial of service, misconfiguration issues, and system level SDN security. To provide protection in SDN architecture and implementation, it is critical to entail the three basic properties of security concept such as confidentiality, integrity and availability of information. This can be achieved by applying authorization, authentication and encryption for a secure operation of the network.

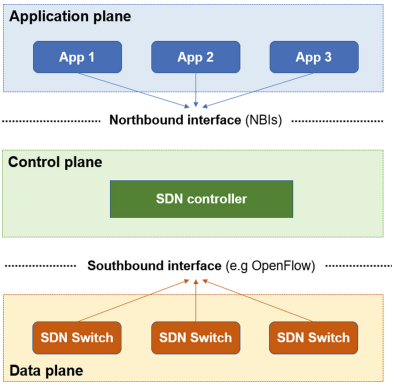

Fig. 1. Architecture of SDN

Regarding to controller position, when the controller is compromised, the information about the entire network, like topology data, flow information, incoming connections or statistics can be disclosed to unintended parties. Even in a bad situation, a whole network can be manipulated in an unauthorized way from the compromised one. In addition, whereas the control-data interface also known as Southbound interface (SBI) is dominant with OpenFlow standard, there is no standard at Northbound interface (NBI) for OF apps locating in the application plane connecting to controller. Many controllers choose RESTful as an implementation for NBI to decouple OF app (which can be run on a different host) and controller. These applications can request information from the controller and external sources, then send instructions to the controller in order to control the network, such as adding or removing flow rules. Due to the lack of standardization, the NBI is less explored than the SBI which has many suggestions for prevention and mitigation of its own vulnerabilities.

When it comes to NBI's vulnerabilities, they all stem from malicious or compromised OF applications. Malicious OF apps can cause data leakage if no control policy is applied in NBI. Whether the application is malicious from the time of its installation, or compromised by an attacker at a future date, has the same effect in these situations.

Once such an application has access to the NBI, the attacker could exploit vulnerabilities of lack of authentication and fine-grained control, then seize full control over the controller to disrupt service or listen in on communication. It can be feasible through hijacking, man-in-the-middle attacks in the communication between controller and network applications where attackers can use malicious apps to compromise controller for illegal actions, according to research of SDN security challenges and countermeasures [3].

Moreover, in the AIM-SDN study [4], Dixit et al. proposed a threat model aiming to identify several vulnerabilities of SDN which heavily relies on its datastores to program and control the network. They conducted several attacks which compromise availability, integrity, and confidentiality of the network by exploiting potential vulnerabilities. In their scope, these attacks stem from a semantic gap problem between different abstractions as part of the SDN network and the datastore design implemented in SDN controllers. Northbound interface API is also described as vital element to be easily vulnerable to many specific attacks on involved SDN entities according to CIA triad of security concept (confidentiality, integrity and availability) for controller. For example, the consistency and accuracy of the information that is stored in the datastores and passed to the network can be manipulated using Northbound or Southbound channels with SDN controller. If the configuration is installed in the network from NBI at the time not primarily intended by the administrator, unauthorized and undesired traffic may be allowed in the network. In addition, unchecked storage and improper management of the stored information could lead to memory overflows and impact the controllers' availability. With their experiments, they realize that an OF app with RESTful privileges could install configuration (flow rules) in the SDN controllers without any responsibility of ensuring validity of rules. There is no limit or threshold of flow rules that an OF app can install, also controllers will always accept a new flow configuration.

Meanwhile, DELTA framework, a SDN-focused security assessment tool for security flaws testing, introduced by Lee et al. [5], having capability of reinstantiating published SDN attacks in diverse test environments. It successfully reproduced 20 known attack scenarios across diverse SDN controller environments and discovered seven unknown SDN application mislead attacks. There are many attacks related to NBI such as Service-Unregistration Attack which enables applications can freely register to parse control messages arriving from the switch. The malicious or compromised applications can dynamically change the services of other applications without constraint, and potentially with malicious intent. Additionally, legitimate applications could be no longer in an ACTIVE state after undergoing Application Eviction Attacks.

As mentioned above, those situations related to vulnerabilities in NBI for communication of OF apps and controller are proven that it is crucial to build an appropriate OF app monitoring mechanism with fine-grained control policy to secure SDN network. Hence, this paper proposes a framework of trust authentication between OF apps and controller to prevent malicious OF app from accessing network resource by fine-grained control policy. By certifying whether OF apps are trusted in their intended activities, our approach can secure a controller from losing of control, information disclosure and a resource exhaustion attack originating from malicious OF app through

NBI. Our framework can operate independently with controller since its plugin architecture and OF app isolation from critical entities in SDN without any significant impact on transparency and functions of application. Trustworthiness of network application is also observed and regulated for applying an appropriate treatment policy into OF app sending commands to controller via NBI. Dissecting the methods of exploiting vulnerabilities in OF apps is out of scope of this research, but the consequences for SDN controller and counter measures are relevant in the coverage of NBI. A mechanism of flow rule conflict detection and prevention is not also discussed here due to our focus on trustworthiness of OF apps via its privileges and API calls.

The remainder of this paper is structured as follows. Section 2 gives an overview of Northbound interface's security issues and the related works. The proposed mechanism of our approach is introduced in Sect. 3. In Sect. 4, the experiments are conducted to analyze the proper working and efficiency of our model. Finally, we draw conclusions and propose some future work in Sect. 5.

2 Background and Related Work

2.1 SDN NBI Security and Trust Management

NBI is a component located in application-control interface, allows OF app interact with controller - the brain of SDN to request and send commands via NBI's APIs. If an application is insecure, or has vulnerabilities, a malicious actor uses such an otherwise legitimate OF app to send harmful instructions to the controller in many ways.

To illustrate security level, threat modeling is used as a procedure for evaluating and optimizing application, network, system by identifying objectives, vulnerabilities and then making counter measures to prevent or mitigate the impacts of threats to the system. According to research [6], there are diverse methods to assess security of a system, such as PASTA, STRIDE, Trike, UMLSec, CORAS,… However, STRIDE model proposed by Microsoft, which is standing for six categories of threats, is considered as the most appropriate methodology for classifying security issues without implementation. Therefore, STRIDE is selected for indicating vulnerabilities in SDN NBI in this research. Table 1 shows the threats categories of STRIDE methodology.

Regarding to Spoofing category, credentials of authentication can be guessed or disclosed by listening in and capturing; then rogue actor can get permission to perform any actions in the controller. Moreover, due to the lack of authentication mechanism in NBI that mandates the OF apps running commands in the network, some harmful or rogue OF apps can take advantage of this privilege and disrupt network operation thereby violating the confidentiality, integrity and availability of the network. Besides, data is transferred between controller and OF app can be altered and tampered with owing to having no encryption technique, according to Tampering vector. Without cryptography-based data transmission, network state information can be disclosed to unintended parties, according to Information Disclosure. In Repudiation category, it is easy for a malicious actor to perform operations anonymously if there is no secure logging policy for sending instructions to controller over NBI. In addition, in the context of Denial of Service, NBI could be unavailable for legitimate OF apps to

properly work if compromised ones torrentially dispatch large amount of traffic or resource-intensive requests to the controller. Finally, relating to Elevation of Privileges type, OF apps should only have the least amount of privileges required for their operations. An OF app with rich-permission can perform actions with serious impacts to the whole network. So, the method of authorization plays an important role in fine-grained control of applications over NBI.

Table 1. Threats categories of STRIDE methodology

Threat category	Security object	Description of reversion
Spoofing	Authentication	Lack of identification in a distinctive way
Tampering	Integrity	Susceptibility to be altered, modified, tampered in whole or in part
Repudiation	Non-repudiation	Not tracking its actions, its events and their role actors
Information disclosure	Confidentiality	Revealing, disclosing its features and communications
Denial of Service	Availability	Resources are partially or totally inaccessible
Elevation of privileges	Authorization	Susceptibility to be accessed by any element without restriction

Additionally, every OF application should be monitored for each behavior with a trust value showing that it is trusted based on its functions and pre-defined permission to communicate with the controller in a trustworthy manner. Thus, attributes of OF apps is required for real-time tracking and adjusting the trust of OF apps working through NBI. All of their historical actions are always under observation and permanently kept for further report about characteristics of behaviors in the network over time. Access control and reliability can be achieved by checking whether an application is trusted or not, in order to enable OF app request, command and consume network resources in the controller.

2.2 Related Work

Despite the gaps in security, though, SDN continues to be an emerging alternative solution to the problems of modern networks. Therefore, the urgency of enhancing security in SDN and protection for the controller are more and more concerned. In this context, several solutions of a secured Northbound interface used for the communication between applications and SDN controller have been proposed. Our work is inspired by prior work in SDN security and vulnerability analysis techniques, particularly regarding NBI security as the follows.

Firstly, a comprehensive analysis of the 22 separate vectors for potential abuse or direct attack that arise from diverse location in SDN is introduced by Yoon et al. [7]. With this research, unauthorized network-view manipulation (Internal Data Storage Modification), unauthorized application management through controller are indicated as some of the NBI vulnerabilities of SDN.

Aliyu et al. [8] proposed a trust management framework for the access of OF apps to the controller. Each OF app's permission is defined in the terminology of four tasks *Read, Write, Notify* and *SystemCall* and this framework takes the responsibility of ensuring no OF app can encroach its granted permission. Moreover, the trust established between applications and controller are also periodically monitored and updated based on their operations.

In [9], Porras et al. developed a kernel solution called FortNOX to overcome the problem of malicious OF app's intrusion to the controller and unauthorized flow rule installation. SE-Floodlight [10] is another solution extended from FortNOX. Both of them categorize applications into three groups and apply authentication based on these roles. However, with the responsibility of processing a large volume of requests, an additional feature of access control and conflict verification can lead the controller to be overloaded and suffered an increasement in response time.

Consider trust as an important factor, Isong et al. [11] introduced a model aiming to protect the controller by requiring a specific OF app to meet at least a trust level to communicate with the controller and consume network resources. This approach creates a trust matrix of applications corresponding to their identity used for effective resources management in SDN.

Besides, ControllerSEPA [12] utilizes a repacking service to manage the transferred data from the controller to OF app, which prevents controllers from being affected by malicious applications. It also provides management services of authentication, authorization, accounting (AAA) based on the features and granted permission of a specific OF app, while these applications communicate with controller via a TLS-enabled northbound interface. Developed as a plug-in model, its operations take place outside of the controller, without significant effect on the overall performance of the SDN control plane will occur.

In comparison with the aforementioned studies, our framework has a proactive approach like a plugin model in ControllerSEPA's idea to enable authentication and access control tasks is performed independently with controller to meet SDN network brain's performance. In addition, trust degree of OF app is also paid attention to monitor for manipulation and adjustment its actions according to the matching between actual behaviors and intended function description with granted permission of network application in SDN. In our trust management principle, a trustworthiness in trust relationship is represented by a trust value which is collected, stored or updated in our framework for further analysis at later time. Then, controller can certify how much trusted an application is to be eligible to execute different actions related to network resource consumption. Moreover, a treatment policy for OF app with assigned trust level is designed to give a relevant counteraction for network administrator, such as blocking or limiting the involvement of OF app with controller through NBI in case of exceeding declaration of actions.

3 Trust Trident – A Trust-Based Authentication Framework

This section presents our proposed framework for trust-based authentication, named Trust Trident, which plays a role as a plug-in intercepting communication between OF apps in application plane and the SDN controller, in order to ensure the security of this centralized component. The name Trust Trident firstly indicates the capability of this framework which provides three services of authentication, authorization, and accounting for security. The architecture of this framework is given in Fig. 2, consisting of four main modules: Authentication module, Authorization module, Accounting module and Repacking service. Additionally, TrustStore and database are used to enable our framework to store trusted certificates and privileges of OF apps when interacting with controller.

3.1 OF App in Context of Trust Trident

In the scope of our research, considered OF apps are applications which communicate with controller for requesting information via its REST API. The plug-in keeps in its knowledge some identity information of a specific application to be able to distinguish one to one.

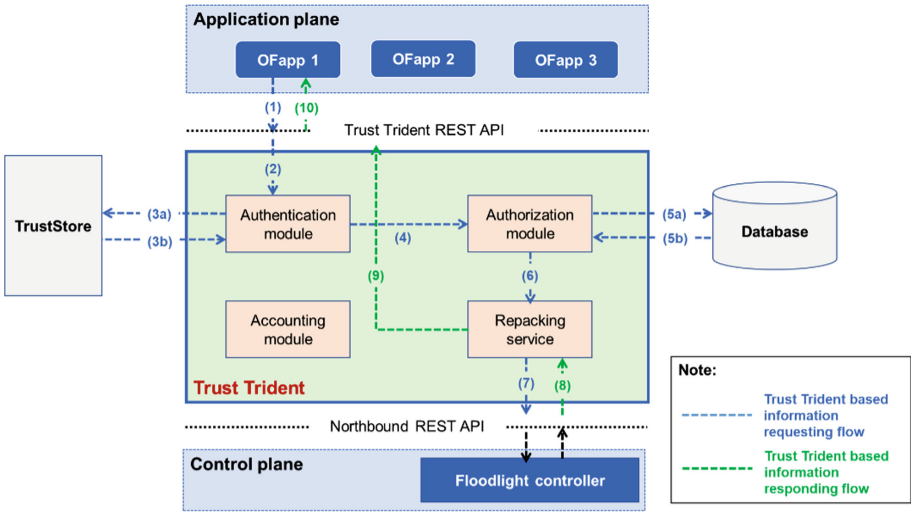

Fig. 2. Trust Trident framework

Certificate. Each application joining in Trust Trident-enabled SDN network must have a certificate for authentication purpose. These certs are pre-created when an application registers itself to the plug-in to gain permission for later requests.

Trust_value. This value represents the measure of trust of our plug-in on a specific application, in the range of 0 to 1000. In our context, the higher *trust_value* is, the more reliable the application is.

Status. An OF app can have its status in *enabled or disabled.* Besides depending on *trust_value* of application, the plug-in can also considers allowing or denying applications from accessing SDN controller based on the current status of it. The administrator can use this property to manually block a potential malicious OF app due to its suspected behaviors.

Permission on a Resource. Inherited from [8], our plug-in defines privileges of an OF app for consuming specific resources in terms of four groups: *READ* for information queries, *WRITE* for modifying or configuring on the network, *NOTIFY* for getting informed about events, *SYSTEM CALL* for requests on system resources.

3.2 Requesting via Trust-Based Authentication Framework

To illustrate, our proposed framework will be dissected in the context of processing a request sent from an OF app to query information from the SDN controller.

Trust Trident REST API. This component, which has similar features like the common Northbound REST API, plays as a communication interface between OF app and our framework. This interface, however, is transparent to applications, OF apps have no doubt and believe that they are interacting with the controller via northbound as usual. With the existing of this interface, traffic is forwarded to our plug-in for authentication purpose before they can get into the target controller.

Authentication Module. In this step, the application is authenticated and verified by the Authentication module. Applications use their pre-created certificates to prove their identity with the plug-in. OF app and our plug-in send its certificates to each other for the bilateral trust to be established. This operation is supported with the third-party TrustStore, where certificates of all component are stored. After that, SSL/TLS (HTTPS) is always used for all of the data transmission between them. By default, any new coming application cannot pass the verification from this module will be blocked until its information is updated in our plug-in.

Authorization Module. This module takes the responsibility of checking whether a request from an application meets the requirements of permission to consume network resources. The verification process takes two steps to complete. At first, our plug-in looks at the *status* of the application to prioritize any manual decision made on it by administrator. A disabled OF app will be denied from accessing network resources regardless of their privileges. In case of active applications, this module makes a mapping of requested information or operation to their corresponding required permissions based on the data stored in the database, then verify if the application has properly granted one to make this request.

This module also keeps an eye on the trust_value of the application in the view of our plug-in. It re-evaluates this value based on the validation of received requests. The regular request will remain the trust in the high value, but over-privilege ones can cause

a drop in trust_value as a punishment, which is 5 points per unprivileged request. Until a pre-defined threshold is met, this module warns the administrator about applications that need more attention.

Repacking Service. Upon the authentication and permission verification process, legal requests need to be sent to controller unchanged to be transparent to OF apps when receiving response data. Our plug-in has this duty performed by Repacking service. Being a traffic forwarder, this component operates like an application in the application plane sending requests to the controller via its Northbound interface, which is REST API in this case. Controller sends a response containing network resources data to applications through Repacking service then.

Accounting Module. Accounting module keeps record of every request sending to the controller, including the application name, the permission of OF app and the *trust_value* of it at the time of that event. This information can be useful in the system audit trail process. Our framework also provides a web interface for administrator to have an overview of underlying operations and receive warning messages of suspicious activities.

4 Experiment and Evaluation

4.1 Network Emulator

To design an experiment to evaluate our proposed mechanism, Mininet [13] is used to demonstrate a simple SDN network managed by a controller. We create a linear-type network with two switches and two hosts connect to each of them. Mininet is run on a Linux virtual machine with Ubuntu 14.04 LTS.

As mentioned above, Floodlight [14] is our choice of SDN controller to develop our solution as its plug-in. Trust Trident is operated on the same machine as Floodlight controller in order to simplify the interception of communication traffic between this core component and OF apps.

Our tested OF apps locate in another machine of Ubuntu, then they connect to controller to request for specific information or features on the managed SDN network.

4.2 Permission and Trust Configuration

For experiment purpose, we pre-define some configurations of mappings between API requests and permissions as well as the trust policy used in our plug-in.

Request and Permission Mappings. In the scope of this paper, we demonstrate and evaluate our proposed plug-in in the case of analyzing some common API requests used by many OF apps. These APIs are categorized into four above-mentioned permissions based on their requested information or operation on the SDN network.

Trust Policy. These configurations are applied based on the *trust_value* of a specific OF app to protect the control plane of the SDN network.

Deactivate Policy. This is the policy for applications with the *trust_value* less than *deactivate trust_value*, which is 50 in our case. Matched applications will be changed to disabled status and no request can be made to the controller during this time.

Low Trust Policy. Applications whose *trust_value* are lower than 75 – the *warning trust_value* are targets of this policy. Legitimate requests from these OF apps are still processed by the controller without any interruptions, but some warning messages will be notified to the network administrator to have more attention on the behaviors of these applications.

High Trust Policy. Applications with decent behaviors will be presented in the high value of *trust_value*, which is classified to the best condition of a given application in the communication with SDN controller. These applications are trusted by plug-in and their operations are considered as harmless to the SDN network.

Default Policy. The policy is applied when no other policy matching its current status.

4.3 Our Scenarios

Four scenarios of OF apps with different pre-granted permissions and requested operations on SDN network will be tested to observe the proper behavior of our plug-in. Our aim is to verify whether the suggested plug-in can catch and analyze these requests for the authentication purpose to prevent controller from communicating with malicious applications. STRIDE methodology is used to evaluate the security object of NBI after running Trust Trident. The information of tested applications is given in Table 2.

Table 2. OF applications used for evaluation

Application name	Granted permission	Requested information/operation	Required permission
Information of SDN	No permission	SDN controller version checking	READ
		SDN topology information	READ
		SDN uptime checking	NOTIFY
SDN Firewall management	READ, WRITE	SDN Firewall status checking	READ
		Firewall rule listing	READ
		Firewall rule configuring	WRITE
SDN Firewall management version 2.0	READ, WRITE	SDN Firewall status checking	READ
		Firewall rule listing	READ
		Firewall rule configuring	WRITE
		SDN uptime checking	NOTIFY

Scenario 1. Authentication of New OF App. A new OF app – called *Information of SDN*, makes its first connection to the SDN network and requests the following information: the version of Floodlight running on the SDN controller, the uptime of SDN and its topology information. This application communicates with the controller in the first time, therefore it has not yet authenticated with the controller as well as our plug-in and no specific permission is assigned to it.

Scenario 2. A Well-Behaved OF App. This is an example of an authenticated application named *SDN Firewall management* with given privileges for specific requests of Firewall status checking, Firewall rule listing and configuring on SDN controller. This application has relevant permissions allowing it to perform these requests without any warnings or impediments from our plug-in. All information transmitted between our plugin and OF app are protected from tampering (targeting on the integrity of data) and information disclosure (targeting on the confidentiality of data) by HTTPS connection.

Scenario 3. Over-Privileged Requests. In case of upgraded applications, some of the new features are added into the current version of applications. *SDN Firewall management* application version 2.0 with some updates compared to the version at scenario 2 can be an example, which introduces a new capability of system uptime monitoring. However, the authentication and authorization of the plug-in have not yet updated this fresh feature and no permission related to it is granted to application.

Scenario 4. Manual Punishment for Consecutive Bad Behaviors. In this test, we observe the decision made by our plug-in in the case of applications trying to request unallowed information or operations for many consecutive times, which alerts the administrator many times about this strange behavior. *SDN Firewall management* is still used as an example in this scenario.

4.4 Evaluation Results

The overall result of our tested scenario can be viewed in Table 3, along with the information about the requests from applications and the corresponding privileges needed for them to be allowed. Comparing these permissions to the granted one of application, we expect some proper operations of our plug-in, also keep an eye on its actual handling tasks to make an evaluation of this proposed solution.

Scenario 1. Authentication of New OF App. Due to the lack of authentication information of this new applications in the knowledge of the plug-in, our solution is not able to identify if this application is a safe one to communicate with controller. Thus, all of its requested information and operation are denied and returned errors as responses. When an OF app desires to use NBI in order to access network resources, it must register with our plugin by declaring which API it wants to access. Subsequently, administrator could consider whether an OF app is granted specific permissions with default *trust_value* or not.

Table 3. Evaluation results

Scenario	Granted permission	Request	Requested permission	Expected operation	Confirmed behavior
(1) Authentication of new OF app	No permission	Controller version checking	READ	Failed to authentication and communicate with controller	Failed to communicate with controller
		Topology information	READ		
		Uptime checking	NOTIFY		
(2) A well-behaved OF app	READ, WRITE	SDN Firewall status checking	READ	All requests are allowed and forwarded to controller	All requests are allowed and forwarded to controller, *trust_value* remains unchanged
		Firewall rule listing	READ		
		Firewall rule configuring	WRITE		
(3) Over-privileged requests	READ, WRITE	SDN Firewall status checking	READ	Requests are allowed and forwarded to controller	Requests are allowed and forwarded to controller
		Firewall rule listing	READ		
		Firewall rule configuring	WRITE		
		SDN uptime checking	NOTIFY	Denied due to over-privilege	Response with error message, *trust_value* is dropped
(4) Manual punishment for consecutive bad behaviors	READ, WRITE	SDN Firewall status checking	READ	Denied due to disabled status of a bad-behavior application	Response with error message, *trust_value* is increased to re-gain the allowed status
		Firewall rule listing	READ		
		Firewall rule configuring	WRITE		

Scenario 2. A Well-Behaved OF App. Requests from example application are legitimate based on its granted permission. As a result, the SDN controller and then our plug-in response it with the corresponding information of Firewall status or the list of Firewall rules, through HTTPS connection. In this context, the integrity and confidentiality are ensured for both parties during bilateral communication. The *trust_value* of this OF app is remained with the current value.

Scenario 3. Over-Privileged Requests. According to the saved authentication information in the plug-in, features of firewall management have been allowed for this application, therefore the result is returned to it without any error. However, the new capability of requesting system uptime is denied, at the same time the *trust_value* of this application is decreased as a punishment for over-privileged behaviors. In the case of the dropped *trust_value* matched the *Deactivate policy*, this application is disabled for further requests.

Scenario 4. Manual Punishment for Consecutive Bad Behaviors. The tested application has sent multiple over-privileged requests, which results in the drop of *trust_value* to 65, therefore administrator has been notified with the number of about 10 warning messages and decides to disallow this application. The punished application, still with a reasonable *trust_value*, is in disabled status so it cannot request even allowed information or operations. However, these abused activities are still recorded in the plugin while the trust of this application can be re-gained manually by the administrator after receiving enough number of legitimate requests in the blocking time.

With mentioned experiments, our framework can secure NBI between controller and OF apps, compliant with STRIDE methodology. Table 4 summarizes the capability of Trust Trident in assuring security properties against different threat categories.

Table 4. Trust Trident evaluation for security assurance in NBI with STRIDE

Threat category	Security object	Solution	Scenario
Spoofing	Authentication	Certificate	1, 2
Tampering	Integrity	HTTPS	2
Repudiation	Non-repudiation	Logging, Accounting	4
Information disclosure	Confidentiality	HTTPS	2
Denial of service	Availability	Out of scope	Out of scope
Elevation of privileges	Authorization	Fine-grained control	3, 4

5 Conclusion

In this paper, we propose a trust-based authentication framework in the NBI, which manages the secure communication between OF apps and SDN controller, to protect and prevent controller - the vital component from being compromised and taken advantages to control the whole SDN network. Our plug-in approach provides the capability of intercepting requests from applications to further analyzing without any interruption or significant effects on operations of the controller. It is attributed to a trust management and fine-grained control activity using privileges. The mechanism of trust aims at giving applications the required privileges to perform in the network and not to exceed declared limit of operation. The experiment results show that this solution is feasible to be used as an effective protection method for controller in many scenarios. In the future, auto-detecting and preventing malicious OF app without the administrator as a supervisor, also integrating mechanism of flow rule verification into this

framework can be a potential research problem in enhancing the security of the NBI as well as SDN network.

Acknowledgement. This work is funded by University of Information Technology, VNU-HCM under grant number of D1-2019-09.

References

1. Kreutz, D., Ramos, F.M.V., Veríssimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. In: Proceedings of the IEEE (2014)
2. Scott-Hayward, S., Natarajan, S., Sezer, S.: A survey of security in software defined networks. IEEE Commun. Surv. Tutor. **18**(1), 623–654 (2015)
3. Li, W., Meng, W., Kwok, L.F.: A survey on OpenFlow-based software defined networks: security challenges and countermeasures. J. Netw. Comput. Appl. **68**, 126–139 (2016)
4. Dixit, V.H., Doupé, A., Shoshitaishvili, Y., Zhao, Z., Ahn, G.-J.: AIM-SDN: attacking information mismanagement in SDN-datastores. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS 2018), pp. 664–676. ACM, New York
5. Lee, S., Yoon, C., Lee, C., Shin, S., Yegneswaran, V., Porras, P.: DELTA: a security assessment framework for software-defined networks. In: Network & Distributed System Security Symposium (2017)
6. Chikhale, A., Khondoker, R.: Security analysis of SDN cloud applications. In: Khondoker, R. (ed.) SDN and NFV Security. LNNS, vol. 30, pp. 19–38. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-71761-6_2
7. Yoon, C., et al.: Flow wars: systemizing the attack surface and defenses in software-defined networks. IEEE/ACM Trans. Netw. **25**(6), 3514–3530 (2017)
8. Aliyu, L., Bull, P., Abdallah, A.: A trust management framework for network applications within an SDN environment. In: Proceedings of 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), Taipei, Taiwan (2017)
9. 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, Helsinki, Finland (2012)
10. Cheung, S., Fong, M., Porras, P., Skinner, K., Yegneswaran, V.: Securing the software-defined network control layer. In: Proceedings of the 2015 Network and Distributed System Security Symposium (NDSS), San Diego, California (2015)
11. Isong, B., Kgogo, T., Lugayizi, F., Kankuzi, B.: Trust establishment framework between SDN controller and applications. In: Proceedings of 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Kanazawa, Japan (2017)
12. Tseng, Y., Zhang, Z., Naït-Abdesselam, F.: ControllerSEPA: a security-enhancing SDN controller plug-in for OpenFlow applications. In: Proceeding of 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), Guangzhou, China (2016)
13. Mininet - An instant virtual network on your laptop (or other PC). http://mininet.org/
14. Floodight Controller - Project Floodlight. https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343514/Tutorials