A Trust Management Framework for Network Applications within an SDN Environment

Aliyu Lawal Aliyu, Peter Bull & Ali Abdallah
Centre for Cloud Computing
School of Computing & Digital Technology
Birmingham City University
Email: (Aliyu.Lawal-Aliyu, Peter.Bull & Ali.Abdallah)@bcu.ac.uk

Abstract—**Software Defined Networking (SDN) is an emerging paradigm that changes the way networks are managed by separating the control plane from data plane and making networks programmable. The separation brings about flexibility, automation, orchestration and offers savings in both capital and operational expenditure. Despite all the advantages offered by SDN it introduces new threats that did not exist before or were harder to exploit in traditional networks, making network penetration potentially easier. One of the key threat to SDN is the authentication and authorisation of network applications that control network behaviour (unlike the traditional network where network devices like routers and switches are autonomous and run proprietary software and protocols to control the network). This paper proposes a mechanism that helps the control layer authenticate network applications and set authorisation permissions that constrict manipulation of network resources.**

Keywords: SDN, Trust, Authentication, Authorisation, Security.

I. Introduction

Software Defined Networking is a new paradigm that separates the control plane (intelligence) from the data plane (forwarding plane) of network devices e.g routers and switches. SDN brings about a major shift in how networks are managed. With SDN, network management moves from implementing high level network functionality with low-level device configurations that are tedious and prone to error [1]. Network functions are instead implemented with software that facilitates automation, programmability, orchestration and debugging[2].

The separation of control and data plane requires the controller to be logically centralised [3], and therefore maintain global knowledge of all the network states, which provides a means of developing more sophisticated network functions like routing, switching, load balancing and intrusion detection/prevention systems [4]. Centralisation brings about performance bottlenecks and single point of failure coupled with security issue if the centralised controller is compromised. However there are proven concepts where the controller implementation is distributed like ONIX [5], SmartLight [6] and ONOS[7] to mitigate the issue of performance bottlenecks and single point of failure. In addition redundancy, fail over, and dependability can be achieved.

The main problem of SDN stems from the benefits it provides [8]. Thus network programmability and control logic centralisation. These introduce new faults and attack plane and give way to new threats that were not present before or difficult to exploit [9]. These new threats are discussed in [10], where a severe threat among them is identified in terms of trust between the network controller and the network application. In SDN environments the controller abstracts the underlying data plane for use by a network application. These network applications are programs written to change and manipulate the state of the network.

There is currently no access control mechanism in place that verifies the interaction and association of network applications with the controller [11]. With the controller being centralised and having full knowledge of the network under control, if a malicious application takes over the controller, the result would be catastrophic. Some of the implications are redirection of sensitive traffic, controller spoofing, DoS attack, network reconnaissance attack, tampering with flow rules etc. So much control is delegated to the controller that its compromise could lead to hijacking the entire network operation [12].

SDN is a new network paradigm at an early stage with the potential of becoming the next generation communication network, much is expected on the side of availability and dependability. Every secure communication network should guarantee confidentiality, integrity, availability, authentication and non-repudiation. This cannot be achieved without having concrete threat mitigation techniques in SDN.

In this paper a trust mechanism is proposed for the network application to interact securely with the controller. Each network application has attributes (behaviour descriptors) and functions it serves within the network. The trust mechanism aims at giving applications the required privileges to run in the network and not to exceed defined limit of operation.

The rest of the paper is organised as follows: Section II explores the background of how network applications interact with the controller. Section III deals with related work in the field of controller to network application security. Section IV provides an insight to the approach used to address the identified gap using attributes in network applications. Section V presents the proposed framework that will be used to establish a secure relationship between the controller and the network application. Finally, section VII concludes and discusses future directions.

II. Background

The SDN paradigm supports third party development efforts and consequently suffers from trust issues on deployed OpenFlow applications. The violation of trust can leads to different types of attacks, where the consequences are severe and have impact on the entire network. Therefore trust violation is a threat and resides between the controller and the application[13]. SDN networks are programmed with policies that explicitly allow network applications to apply changes in the network and no formal verification technique or semantics to assess the trust of these applications[14]. Malicious applications can abuse these privileges and harm network operation.

When applications are instantiated with the controller, they automatically inherit all the access rights to change, manipulate or modify network state. This is a serious threat that receives little attention and sparsely explored within literature. The threat posed by third party applications arises from how to verify the trustworthiness and reliability of a program module, because network applications have access to critical network resources like flow table, device configurations, memory, input and output port.

Most network administrators that use third party applications assume same trust level similar to that of the controller on the network applications [13]. This is because controller modules must undergo series of tests and verification to make sure that they are reliable and fit for use in a production network. However it is difficult to ascertain the reliability and trustworthiness of a third party application. A malicious or compromised application can be a sink for various network based or host based exploits. This can give way to control plane attacks that can lead to code execution or information disclosure.

Figure 1 shows a bug that resides between the controller and the network application [10]. If there is no mechanism or framework that authenticates the application when running scripts in the network, some malicious application can take advantage of this privilege and disrupt network operation thereby violating the confidentiality, integrity and availability of the network.

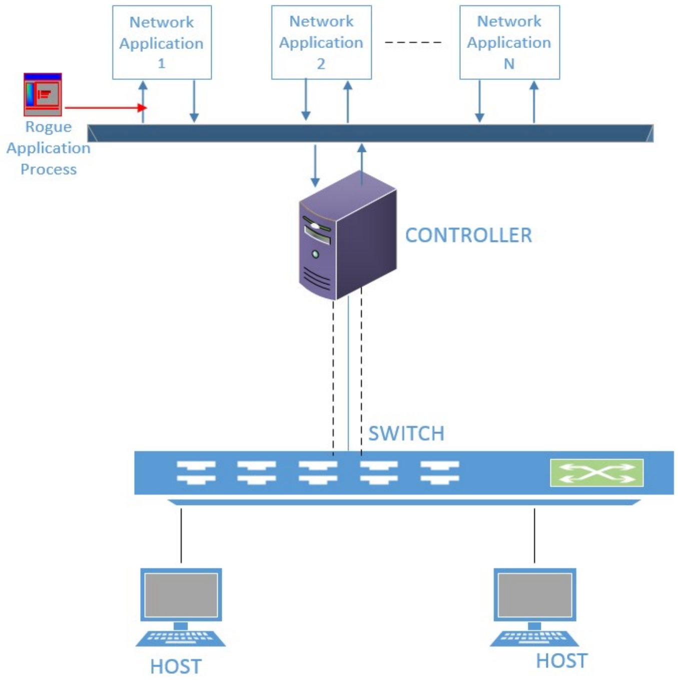

Fig. 1. Application to Control layer threat

III. Related Work

[15] implements a layer which sits between the controller and data plane equipment called Trusted Oriented Controller Proxy (ToCP) . The ToCp is responsible for comparing flow rules from different controllers and installing the most trusted one on the data plane devices. Their idea entails the use of several redundant controllers connected to the ToCP, and the ToCP gathers and analyses network configuration requests from different controllers. And if ToCP trusts the request it will then enforces it on the respective data plane devices.

This approach is more of controller to data plane security rather than trust between controller and network application. Nevertheless the ToCP monitors and analyses request coming from network applications emanating from different controllers. One of the limitation of ToCP is the added complexity in accepting requests from multiple controllers. This brings latency issues in the mechanism which makes it not to scale and suffer from performance issues as traffic increases in the network. Distributed controller implementation brings issues of timing, synchronisation, consistency and coordination[6]. ToCP lacks mechanism that checks the trustworthiness of the network application itself not the configuration sent by the network application.

Rosemary[16] is an SDN controller that is designed and centred on security. It aims at providing control layer resiliency in the event of a network application crashing, instabilities or vulnerabilities. In some situations the failure of a network application may halt the operation of the control layer. Rosemary achieves this through application spawning that provides containment and resiliency. Rosemary does not provide trust between application and controller, but provides a means of threat containment in case of malicious application compromise. It has the capability to isolate the fault and keeps the network operating system functional.

Another controller that provides security at the control layer is the Security Enhanced Floodlight (SE-Floodlight)[17]. The control layer of the Floodlight controller is extended to provide additional security features and tackles rule conflicts in environments where multiple network applications are deployed. The Northbound Application Programming Interface (API) is used to provide the isolation between controller and application process [18]. However, the Northbound API has a weak authentication mechanism and without a defined trust model for messages in transit [19]. The API lacks appropriate authorisation mechanisms and can be taken over by a malicious application. SE floodlight does not provide a trust model that can be used to establish a trusted relationship between controller and the respective network application.

The ability to segregate and isolate the different applications running on the controller in order to provide logical segmentation to support authentication of the applications and to enforce level of authorisation and privileges will be paramount to a secure and dependable control layer.

IV. Trust Attributes

Every application that runs and make changes to network configurations through the controller has attributes[20]. These attributes are the building blocks and the main function of the network application. Through these attributes the controller can implement the changes to the underlying data plane devices. Since the attributes are the point of concentration that execute the logic in the network, the controller can utilise the various attributes and establish an access control mechanism based on trust that can authenticate and authorise this attributes in the network. The attributes are functional descriptors that define the several aspect of the network application.

There is no defined trust mechanism that helps establishes a trusted relationship between the control and the various network applications[10]. These attributes are key in determining the required privileges needed by a particular network application to run on the network. Through the attributes, limits and threshold can be set so that applications cannot use network and computing resources beyond the allocated portion.This resources include memory space, file space, CPU execution time etc. And should execute within defined finite time and set boundaries.

With the help of the attribute the controller can track application behaviour and monitor the activities of the application. Through this, security and reliability can be achieved by checking whether an application is trusted or not. This paper proposes a trust model that checks the various attributes of an application and provide an authentication and authorisation privileges for network applications to run securely with the control layer.

A. Access Control

The attributes carry out the desired goals of the network application. Applications do not have a limit to how many attributes they can have to control network function. It is the responsibility of the control layer to limit the operation and execution of these attributes in order to have a controlled and secured environment. The four main permission access an application can have to exercise control over network applications are Read, Write, Notify and System calls [13][21].

1) Read: This allows the application to learn about the state of the network including topology, location of servers and end hosts, reading flows, statistics of events, read packet payload etc [22]. Having this permission allows the application to map the network and track activities. Malicious applications can carry out a reconnaissance attack just with the *Read* permission.

It aids the attacker in knowing the next line of action since network events and states are learned via the read permission. There is a huge state space in just the *Read* permission, it extends to learning how many switches are communicating with the controller and the respective flow table entries, buffer and memory space. Sensitive OpenFlow information can be obtained via the *Read* permission.

2) Write: This gives OF applications the ability to modify certain states of the controller and the switch. It includes dropping flows, forwarding of flows to output ports, set VLAN tags, set ques for QoS etc. The write permission allows application to implement changes in the network. Write permission gives application the ability to tamper with various state space of events in both the controller and the switch. For a rogue application to have this permission, the consequences will be severe because flow entries can be rerouted, deleted, forwarded to another malicious for inspection etc. The impact is severe with disruption or taking over the control of the entire network.

3) Notify: Notify permission delegates when an OF application should receive notification of events as they occur in real time or on-demand. Such notifications that can be received are switch join or leave event, switch port status, link status , total number of flows received or transmitted on a port etc. Applications can subscribe to events and get updates, a malicious application can subscribe to a certain event and monitor the activity example whether server is active and on which port number the communication is taking place. It can be used in tracking whether certain services are present in network.

4) System Call: This manages how an application makes request to operating system resources. These resources include CPU, RAM, Input and Output. The system comprises of the code base on which the controller sits. Applications with this permission can reserve memory and use disk space that is supposed to be shared with other native applications.

A potential memory depletion attack can be launched with this permission. An application can span it processes across the various available cores of CPU for more than it required time of execution making it difficult for other applications to complete their task with the available processor resources.

The four permissions *Read*, *Write*, *Notify* and *System Calls* can be viewed as access tokens that if an application has unlimited access to, then controlling the entire network will be easy. The proposed framework is aimed at constricting these access permission to an application by narrowing the required access needed only by the network application to run. An application should not have full access to network resources because a malicious application can drop event, put the system into infinite loop, overwrite memory and take over network control.

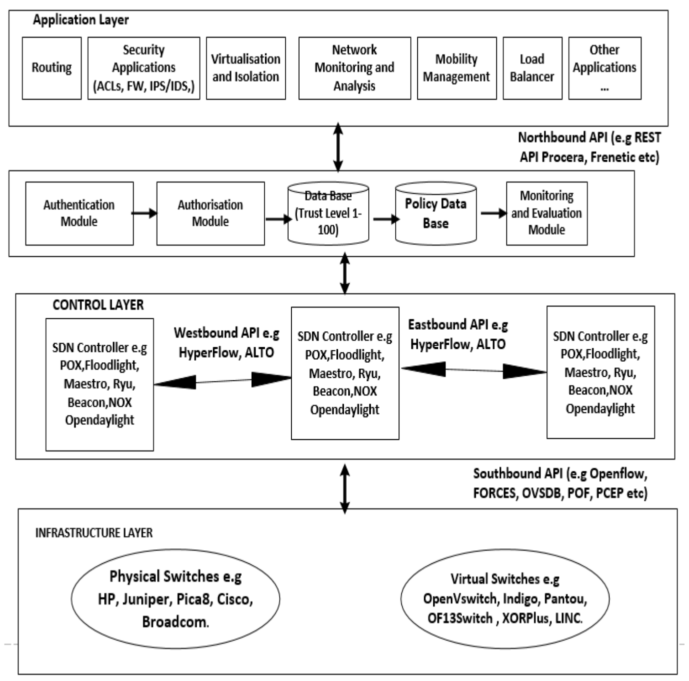
Application Layer
Routing
Security Applications (ACLs, FW, IPS/IDS,)
Virtualisation and Isolation
Network Monitoring and Analysis
Mobility Management
Load Balancer
Other Applications ...
Northbound API (e.g REST API Procera, Frenetic etc)
Authentication Module
Authorisation Module
~~Data Base~~ (Trust Level 1-100)
Policy Data Base
Monitoring and Evaluation Module
CONTROL LAYER
SDN Controller e.g POX,Floodlight, Maestro, Ryu, Beacon,NOX Opendaylight
Westbound API e.g HyperFlow, ALTO
SDN Controller e.g POX,Floodlight, Maestro, Ryu, Beacon,NOX Opendaylight
Eastbound API e.g HyperFlow, ALTO
SDN Controller e.g POX,Floodlight, Maestro, Ryu, Beacon,NOX Opendaylight
Southbound API (e.g Openflow, FORCES, OVSDB, POF, PCEP etc)
INFRASTRUCTURE LAYER
Physical Switches e.g HP, Juniper, Pica8, Cisco, Broadcom.
Virtual Switches e.g OpenVswitch, Indigo, Pantou, OF13Switch , XORPlus, LINC.

Fig. 2. Framework

V. Proposed Framework

The proposed framework aims at establishing a trusted relationship between the control layer and the network applications. This is achieved by authenticating the various network applications and setting authorisation rules (privileges) as to how network application can use network resources. Five modules are introduced in the control layer. They are:

- Authentication Module
- Authorisation Module
- Trust Database
- Policy Database
- Monitoring and Evaluation Module

Figure 2 presents the framework in SDN architecture, the application layer hosts the various network applications such as routing, firewall, load balancer etc. The control layer is the logic via which network elements in the infrastructure layer are controlled. The infrastructure layer contains combination of either physical or virtual forwarding elements e.g switches.

1) Authentication Module: Authentication is the first phase in the framework and is triggered when an application initiates a request to implement network changes via the control plane. The control plane gets the request and starts the challenge request for the credentials of the application. The application responds back with the challenge response providing the queried credentials and the controller further check the authenticity of the application by consulting the data store in Trust Database. Upon successful verification and validation of the application credential, the controller either grant or deny the application access to network resources.

Every network application that runs is associated with attributes. These attributes are the building block of the application. Through these attributes the controller can implement the changes to the underlying data plane devices. The concept is to use the attributes as subjects, so that they will serve as credentials to verify the identity of the application.

The idea is to know the correct credentials to pair in identifying the applications. With authentication, malicious application cannot get access to the control plane because each application has to be verified before granting access to the network.

2) Authorisation Module: This deals with the specific permissions the application is allowed to execute on the network after successful authentication. Permissions are stored in the Trust Database and the authorisation module consults the database for assigned privileges to applications. The main permissions are *Read*, *Write*, *Notify* and *System Calls*. There is huge state space of control delegated to these four permission access. Applications should be given minimum access required to carry out their functions.

For instance a monitoring application that report ports events and gather statistics about link status should not be given access to system calls. Because that is not part of its primary function, so any suspicious request to *system calls* from that application should be denied and refused in the future.

3) Trust Database: This is the data store that both the authentication and authorisation modules consult. In addition trust values are assigned to various networking applications. The value assigned is subjective and based on the assessment of the network administrator. The trustworthiness is assigned after successful authentication of the network application. After which the trust value is reviewed periodically, based on the behaviour of the application.

Trust takes various values of trust (distrust) which can be represented in a numerical range (0 - 100) or hierarchical order e.g high, medium or low. The database is the most sensitive asset in the framework, as such adding extra layer of security is paramount. Some of the possible security measures that can be applied are containerisation[23] or replication [6].

4) Policy Database: This is the module that defines the implementation of the global network policy. It governs the implementation of the access policies. Example:

i).If the application fails authentication:

Action $\rightarrow$ Deny or forward to administrator.

ii). If an application is explicitly allowed by an Administrator:

Action $\rightarrow$ Grant access with warning.

5) Monitoring and Evaluation Module: The relationship that exists between controller and the network applications needs to be reviewed and evaluated periodically. This is due to dynamic nature of SDN environment, if there is a variation, deviation or anomaly from the main function of the application then the relationship between the controller has to be reviewed. Either the application is quarantined and revert to normal operational behaviour or the application will be flagged and considered rogue. There are instances where trusted and authenticated application can be malicious, but with granular monitoring full visibility of the application behaviour can be observed and reports can be sent to a secured monitoring server or station for further analysis.

The framework aims at improving the security and reliability of the control layer. Control layer resiliency is key to uninterrupted network operation. The transition of traditional network to all SDN network is hampered by many security threats and vulnerabilities out of which trust violation stands out as the most severe. .

VI. Evaluation

Attributes are behaviour descriptors and they vary from one application to another. They form the building blocks of an application and they can be used as subjects when authenticating and authorising an application. Two network application are selected to assess the feasibility of the proposed framework. They are :

1) IP Blacklist Application
2) Learning Switch Application

Each application will go through the framework modules, the implemented modules decide whether an application is trusted and verified to carry out network operation. Two scenarios will be presented where on the first instance an application is permitted with only the required access and the second scenario the application is denied.

A. IP Blacklist Network Application

This application denies access to certain IP addresses that are considered malicious or threat to sensitive network resources. The application works by installing flows on edge switches via the controller, these flows instruct the edge switches to forward any Domain Name System (DNS) request to the controller. When there are matching flows with this criteria the switches forward the DNS queries to the controller, upon receiving these queries by the controller. The controller consults a database containing malicious and harmful IP addresses, if a match is found then that request is denied else the query will go successfully without any interruption.

1) Scenario: To test the application against the framework, the following scenario is adapted to show how the control layer is secured against unauthorised access. The blacklist application when initiated will go through the Authentication Module, this module will verify the authenticity of the application by sending challenge request to the application, upon receiving the right credentials from the application access is granted and Authorisation permissions are assigned to the application. These permissions are set before hand in the Authorisation Module for every application and are subjective based on the assessment of the network administrator.

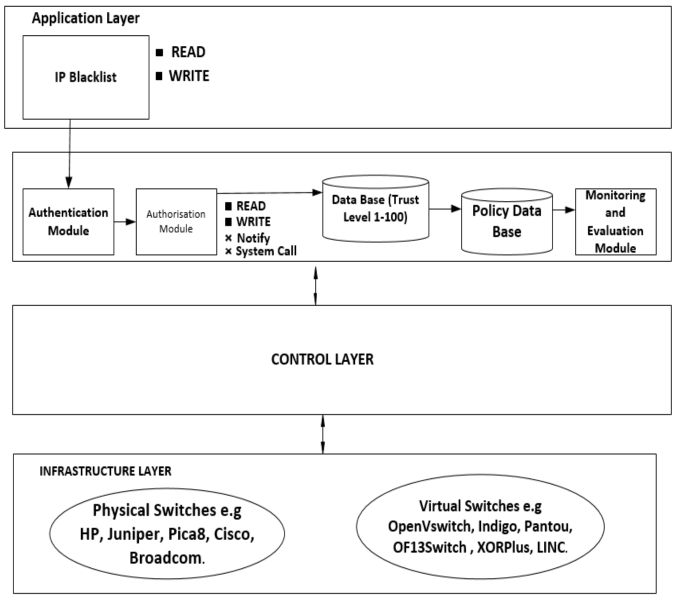
Application Layer
IP Blacklist
■ READ
■ WRITE
Authentication Module
Authorisation Module
■ READ
■ WRITE
× Notify
× System Call
Data Base (Trust Level 1-100)
Policy Data Base
Monitoring and Evaluation Module
CONTROL LAYER
INFRASTRUCTURE LAYER
Physical Switches e.g HP, Juniper, Pica8, Cisco, Broadcom.
Virtual Switches e.g OpenVswitch, Indigo, Pantou, OF13Switch , XORPlus, LINC.

Fig. 3. IP Blacklist Scenario

The blacklist application has only two permission set which are (read and write) due to the nature of its behaviour and functionality. The read permission allows the application to make sense of the presented IP addresses which are checked against a backend database, upon verification the application write flows to the edge switches whether to drop or to allow the communication.

It would be an anomaly for the application to make system calls or receive notification about link status or switch join/leave information. Because the permissions *Notify* and *System Call* are not enabled for this application. Any contrary or strange network access by the application would be monitored and access would be denied if there is deviation. This functionality is implemented by the monitoring and evaluation module, the framework assigns trust level to the application based on how consistent and uniform the behaviour of the application is. Network application can start in a trusted domain and end up in the untrusted domain due to changing behaviours and anomalies. Figure 3 depicts the process as described.

B. Learning Switch Network Application

This application maintains a mapping of host MAC address and associated switch port. The application learns the location of the host when packets are received from the switch port. This helps the switch keeps a lookup table that binds the source MAC address and the incoming switch port. Upon receiving a packet the switch carries out a lookup operation of the packet destination MAC address, if the destination address is contained in the table the switch forwards it to the respective port otherwise the switch add that port in its MAC address table and flood the packet to all ports except the one the packet was received.

1) Scenario: When the learning switch application is instantiated, the controller queries for application credential. The application supplies the credentials (subjects which are part of the identified attributes from the application). Upon receiving the credentials the controller runs a quick check against the database to match the presented credentials, in this case the learning switch application failed to provide the matching credentials and as a result access is denied. The responsibility of the framework is to make sure unauthorised applications are denied access to sensitive network resources.

Third party applications are on the rise in SDN environment and with the advent of cloud computing where infrastructures are virtualised and shared among many tenants a remote adversary can install application in your container or virtual partition. The resulting effect would be severe if there is no framework that verify the authenticity of network applications that make network changes.

VII. Conclusion & Future Work

The proposed trust framework aims at addressing a vulnerability in the SDN architecture that exists between the controller and the network applications. The SDN paradigm supports third party application deployment and consequently suffer from trust issues on implemented Open-Flow applications. The violation of trust can lead to different types of attacks and the consequences are severe and heavily impact the entire network operation. In SDN environment, networks are programmed with policies that explicitly allow network applications to implement changes in the network and malicious applications can abuse these privileges and harm network operation. The proposed framework introduce modules that verify the authenticity of the network applications and assign privilege permissions. The framework monitors verified applications overtime and re-evaluate controller to application trust relationship in case an anomaly is detected. Future work will look into implementing the framework on testbed and report about reliability, performance implication of the various stages of verification and the scalability of the proposed framework.

References

[1] R. Ahmed and R. Boutaba, "Design considerations for managing wide area software defined networks," *Communications Magazine, IEEE*, vol. 52, no. 7, pp. 116–123, 2014.

[2] Y. Jarraya, T. Madi, and M. Debbabi, "A Survey and a Layered Taxonomy of Software-Defined Networking," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 1–29, 2014.

[3] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, "Fabric: a retrospective on evolving SDN," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 85–90.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[5] 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." in *Proceedings of the 9th USENIX conference on Operating systems design and implementation OSDI*, vol. 10, 2010, pp. 1–6.

[6] F. Botelho, A. Bessani, F. M. Ramos, and P. Ferreira, "On the design of practical fault-tolerant SDN controllers," in *2014 Third European Workshop on Software Defined Networks*. IEEE, 2014, pp. 73–78.

[7] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed SDN OS," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6.

[8] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *IEEE Journal & Magazine*, vol. 103, no. 1, pp. 14–76, jan 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6994333

[9] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, pp. 151–152, 2013.

[10] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, pp. 55–60, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2491185.2491199

[11] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for sdn? implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, 2013.

[12] S. Shin, P. A. Porras, V. Yegneswaran, M. W. Fong, G. Gu, and M. Tyson, "Fresco: Modular composable security services for software-defined networks." in *NDSS*, 2013.

[13] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, "Towards a secure controller platform for openflow applications," *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, p. 171, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2491185.2491212

[14] M. Canini, D. Venzano, P. Perešíni, D. Kostić, and J. Rexford, "A nice way to test openflow applications," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 2012, pp. 127–140.

[15] S. Betge-Brezetz, G. B. Kamga, and M. Tazi, "Trust support for SDN controllers and virtualized network applications," *1st IEEE Conference on Network Softwarization: Software-Defined Infrastructures for Networks, Clouds, IoT and Services, NETSOFT 2015*, pp. 0–4, 2015.

[16] S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, and B. B. Kang, "Rosemary: A robust, secure, and high-performance network operating system," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. ACM, 2014, pp. 78–89.

[17] P. A. Porras, S. Cheung, M. W. Fong, K. Skinner, and V. Yegneswaran, "Securing the software defined network control layer." in *NDSS*, 2015.

[18] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A Survey of Security in Software Defined Networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 1–33, 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7150550

[19] A. Feghali, R. Kilany, and M. Chamoun, "SDN security problems and solutions analysis," *2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*, pp. 1–5, 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7293514

[20] 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*. ACM, 2012, pp. 121–126.

[21] A. Ferguson, a. Guha, and C. Liang, "Participatory networking: An API for application control of SDNs," *Sigcomm*, pp. 327–338, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2486003

[22] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: towards an operating system for networks," *SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[23] A. Manu, J. K. Patel, S. Akhtar, V. Agrawal, and K. B. S. Murthy, "Docker container security via heuristics-based multilateral security-conceptual and pragmatic study," in *IEEE International Conference on Circuit, Power and Computing Technologies (ICCPCT), 2016*, 2016, pp. 1–14.