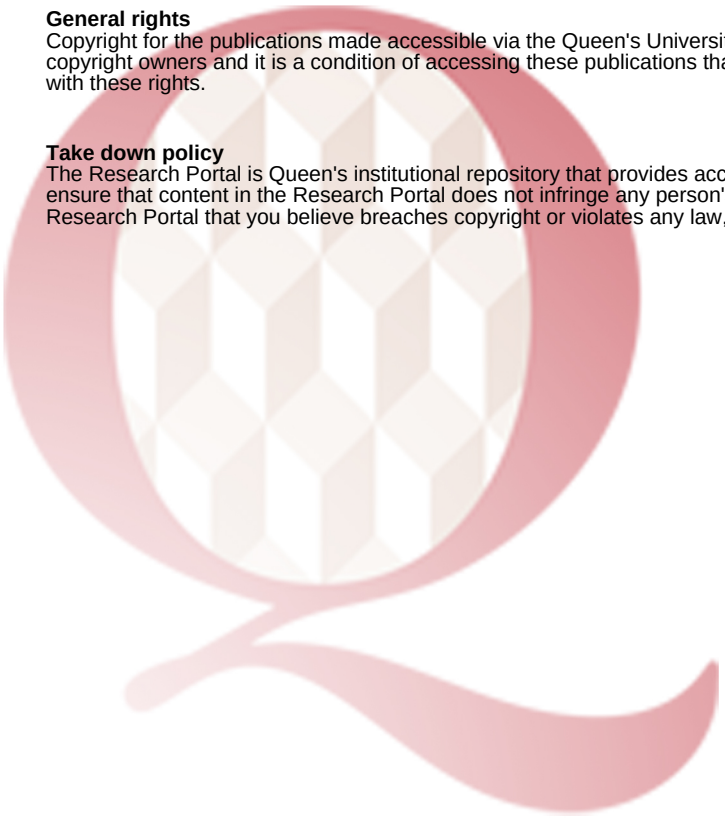
Design and deployment of secure, robust, and resilient SDN Controllers

Scott-Hayward, S. (2015). Design and deployment of secure, robust, and resilient SDN Controllers. In 2015 1st IEEE Conference on Network Softwarization (NetSoft). Institute of Electrical and Electronics Engineers (IEEE). DOI: 10.1109/NETSOFT.2015.7258233

Published in:
2015 1st IEEE Conference on Network Softwarization (NetSoft)

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal



Design and deployment of secure, robust, and resilient SDN Controllers

Sandra Scott-Hayward
Centre for Secure Information Technology (CSIT), Queen's University Belfast, Belfast, BT3 9DT, N. Ireland
Email: s.scott-hayward@qub.ac.uk

***Abstract*—The scale of the Software-Defined Network (SDN) Controller design problem has become apparent with the expansion of SDN deployments. Initial SDN deployments were small-scale, single controller environments for research and use-case testing. Today, enterprise deployments requiring multiple controllers are gathering momentum e.g. Google's backbone network, Microsoft's public cloud, and NTT's edge gateway. Third-party applications are also becoming available e.g. HP SDN App Store. The increase in components and interfaces for the evolved SDN implementation increases the security challenges of the SDN controller design. In this work, the requirements of a secure, robust, and resilient SDN controller are identified, state-of-the-art open-source SDN controllers are analyzed with respect to the security of their design, and recommendations for security improvements are provided. This contribution highlights the gap between the potential security solutions for SDN controllers and the actual security level of current controller designs.**

I. Introduction

The controller or network operating system (NOS) is at the heart of the Software-Defined Network (SDN). A multitude of SDN controllers have been designed since the development of the OpenFlow protocol [1] for communication between the data and control planes of the SDN. A detailed classification of SDN controllers is provided in [2].

Early controller design focused on building support for the OpenFlow (OF) Application Programming Interface (API) i.e. as an OF driver in order to provide a set of basic network services such as link discovery, topology generation and flow entry pushing. In order to meet the performance of traditional (non-SDN) networks, the focus then turned to performance and scalability. Performance enhancements aimed to achieve a high throughput and low latency in terms of control event processing. Scalability refers to the controller's ability to support an increasing number of switches/hosts.

From a security and reliability perspective, the initial concern was the centralized controller as a single point of failure in the network [3]. If all the control decisions take place in one device and an attacker hi-jacked that device, it would take control of the network. Similarly, the centralized controller becomes vulnerable to Denial-of-Service (DoS) attacks [3].

With the evolution of SDN, the vulnerability of the centralized controller is no longer the key security issue. Some issues of concern are the integration of 3rd party applications with the SDN control framework, policy conflict resolution between multiple applications in the SDN, and the complexity of multiple controller deployments. There is no single SDN controller that currently delivers security, robustness, and resilience in parallel. By secure, robust, and resilient (referred to here as 'security'), it is meant that the controller is designed to reduce the risk of intrusion/attack at the network control layer, is able to withstand errors in control layer logic, and is able to recover quickly from disruption and maintain an acceptable level of service in the face of faults.

Five controllers have been selected for security analysis; three of the latest, advanced open-source SDN controllers (OpenDaylight (ODL) [4], Open Network Operating System (ONOS) [5], and Ryu [6]), and two research-driven, security-focussed controllers (SE-Floodlight [7] and ROSEMARY [8]).

The paper is organized as follows: Section II introduces related work. In Section III, the selected controllers are presented. In Sections IV, V, and VI, Secure Controller Design, Secure Controller Interfaces, and Controller Security Services are discussed. Recommendations for future security enhancements in SDN controller design are provided in Section VII. Section VIII concludes the paper.

II. Related Work

Several comparative analyses of SDN/OF controllers have been performed. In [11], the authors presented a study of SDN OF controller performance. However, the controllers (NOX-MT, Beacon and Maestro) have since been superseded by other controllers. In [12], several open-source SDN controllers were compared in terms of performance, scalability, reliability, and security. The security analysis is limited to consideration of how the controllers in the study respond to malformed OF messages. A feature-based comparison of controllers was presented in [13]. Ten selection criteria were used to select the best controller based on an adapted Analytic Hierarchy Process (AHP). Pairwise prioritization of the criteria as used in the AHP is subjective. Ryu is selected to be the best controller.

Two of the controllers discussed in this work are designed for security; ROSEMARY [8] and SE-Floodlight [7]. In addition to these secure controller designs, individual features of the secure control layer framework have been studied. In [14], a means to secure the control channel in software-defined mobile networks based on Host Identity Protocol (HIP) is presented. A fault-tolerant controller architecture with a data store based on a replicated state machine is presented in [15]. With LegoSDN [16], the authors consider the impact of SDN application failures on the controller reliability. In *PermOF*

TABLE I
TEST CONTROLLER VERSION DETAILS

Controller	Source	Version	Release	Architecture	Objective	Security Features
ONOS [5]	ON.Lab	Avocet 1.0.0	2014	Distributed	High-availability, Scale-out, Performance [9]	Security-mode ONOS proposed for v2
OpenDaylight (ODL) [4]	OpenDaylight Project	Helium (Karaf 0.2.0)	2014	Distributed	Enterprise-Grade Performance, High Availability	AAA Service, Foundation of Security Group [10]
ROSEMARY [8]	KAIST, SRI International	-	2014	Centralized	Robust, secure, and high-performance NOS	Process Containment, Resource Usage Monitoring, App Permission Structure
Ryu [6]	NTT	3.13	2012	Centralized Multi-Threaded	High quality controller for production environments	Secure control layer communication
SE-Floodlight [7]	SRI International	Beta 2	2013	Centralized	Security-enhanced version of Floodlight controller	Security enforcement kernel (AAA)

[17], the authors propose a set of application permissions and an isolation mechanism to enforce the permissions.

In comparison with [11]–[17], this work considers a unique set of the most advanced open-source SDN controllers, a broad set of security issues and the design and deployment features to overcome these issues.

III. CONTROLLERS AND TEST ENVIRONMENT

The controllers have been selected based on their design. ONOS and OpenDaylight (ODL) are designed for scale-out i.e. multiple distributed controller instances. ROSEMARY and SE-Floodlight have been designed for security, and Ryu is included based on both security features and extensibility. Although, ROSEMARY is a conceptual design with no source code available, it is considered important in this analysis for the advanced security features introduced in the design.

The security features of the controllers are identified based on both documentation and verification of code (except ROSEMARY). Tests were carried out in a physical SDN with Dell Precision T3600 workstations running Ubuntu 14.04 LTS. The controllers are connected to a network of OpenvSwitch (OVS) v2.3.0 OF switches installed on the Dell workstations with VM hosts. The controller versions are detailed in Table I.

IV. SECURE CONTROLLER DESIGN

The security attributes identified as necessary for the design and deployment of a secure, robust, and resilient SDN controller have been categorized into three groups; Secure Controller Design, Secure Controller Interfaces (Section V); and Controller Security Services (Section VI). An overview of how these attributes are supported by the individual controllers is provided in Table II.

A. Control Process (Application) Isolation

Control process isolation is defined here as the ability to separate the application processes running at the controller in order to provide logical segmentation, to support authentication of individual applications and to apply levels of authorization dependent on trust. Application isolation should provide resilience to the controller whereby a failure/error in one application does not compromise the controller.

Apache Karaf is the OSGi framework used in ONOS and ODL Helium. This enables applications (as bundles) to be dynamically loaded/unloaded. However, this does not provide protection at the level of the control processes that support the applications and there is currently no OSGi application security for access permissions on the Karaf features. This is identified as a future improvement in ONOS and ODL.

The central objective of ROSEMARY is to improve the resilience of the control plane to buggy and malicious applications. To achieve this, the authors propose a micro-NOS architecture. Each OF application is run within an independent instance of ROSEMARY effectively sandboxing the application to protect the control layer from any vulnerability or malicious operation of the application.

SE-Floodlight uses the northbound API to separate the application and control processes. Privilege-based OF operations are enforced with privileges defined by Role and Group Access Control Lists and credential files for internal java-based Floodlight modules. Groups define permissible OF operations.

As demonstrated by ROSEMARY and SE-Floodlight, a sandbox/containerized approach combined with strong authentication and identification services is required to provide protection to the controller from malicious/buggy applications.

B. Implementation of Policy Conflict Resolution

The problem of policy conflict presents itself when the controller receives incompatible flow rules from 2 or more applications. Several solutions to the issue of network policy conflict arising from multiple applications in the SDN have been proposed, as discussed in [3]. However, a concern with these methods is their scalability for large applications/networks with regular flow rule additions/updates. In each case, extensive processing is required in order to detect potential network state misconfigurations. A potential alternative that does not sacrifice performance is proposed in [15]. A strongly consistent data store maintains network state consistency.

Of the 5 controllers analyzed in this work, only ONOS and SE-Floodlight implement policy conflict resolution. In ONOS, the application describes its network requirements in the form of "intents" and ONOS translates these intents with respect to the network configuration. This is supported by a shared

TABLE II
SECURITY ATTRIBUTES OF SDN CONTROLLERS

Controller	ONOS	OpenDaylight (ODL)	ROSEMARY	Ryu	SE-Floodlight
IV. Secure Controller Design					
- A. Control Process (Application) Isolation	x	x	✓(micro-NOS)	x	✓(Privilege-Based)
- B. Implementation of Policy Conflict Resolution	✓(Data-Store)	x	x	x	✓(Algorithm)
- C. Multiple Controller Instances - Resilience	✓(Clustering)	✓(Clustering)	x	x	x
- D. Multiple Application Instances - Resilience	x	x	x	x	x
- E. Secure Storage	✓	✓	✓	✓	✓
V. Secure Controller Interfaces					
- A. Secure Control Layer Communication	x	✓(D-CPI)	x	✓(D-CPI)	✓(D-CPI, A-CPI)
- B. GUI/REST API Security	x	✓(weak)	n/a	x	x
VI. Controller Security Services					
- A. IDS/IPS Integration	x	✓(Defense4All)	x	✓(Snort)	✓(BotHunter, Sec. Actuator)
- B. Authentication and Authorization	x	✓	✓	x	✓
- C. Resource Monitoring	x	x	✓	x	x
- D. Logging/Security Audit Service	✓	✓	✓	✓	✓

data store, similar to [15]. In SE-Floodlight, the SEK uses an algorithm called *Rule-chain Conflict Analysis (RCA)* to detect when a new flow rule conflicts with one or more rules already present in the switch flow table, as described in [7].

C. Multiple Controller Instances - Resilience

The requirement for multiple controllers in SDN was first raised as a solution to the issue of the centralized controller. In order to provide control layer resilience, solutions evolved from simple controller replication schemes to distributed control system design. Distributed design introduces issues of timing, consistency, synchronization, and coordination.

A key feature of ONOS is its distributed architecture to support scale-out and fault tolerance. Multiple ONOS instances can be linked to form an ONOS cluster with each instance the exclusive master of a set of switches. In the case that one ONOS instance fails, the remaining instances elect a new master for each of the affected switches. State management is provided by RAMCloud datastore and Zookeeper registry. ODL supports a similar clustering model using Infinispan NoSQL datastore. For state synchronization, the Cache data structure is replicated in each cluster node. Neither ROSEMARY nor SE-Floodlight consider multiple controller instances. With Ryu, cluster support for distributed deployment is future work.

D. Multiple Application Instances - Resilience

With multiple controller instances, the provision of multiple application instances must also be considered. For applications using OSGi for controller communication (e.g. ONOS, ODL), the application must ensure its own resiliency in the case of failure of one controller instance in a cluster. The application is responsible for providing state synchronization between instances. In contrast, for REST applications, the application-controller connection is non-persistent such that in the case of failure of one controller instance in a cluster, a new connection is simply established on the next transaction. None of the studied controllers provide a mature solution for handling network state coordination across multiple app instances.

E. Secure Storage

The controller contains valuable network state information that must be secured. In a review of the controllers, standard security practices are applied e.g. default permissions on log files to allow owner read/write privilege but read-only to others. Additional measures are applied to the individual controllers. For example, in ROSEMARY, access to internal storage modules is controlled so that applications are authorized for certain operations. Data structures have privileges set and access is dependent on the assigned privilege. Although admin/owner security permissions are set for log files and network databases, in deployment scenarios, it is recommended to consider further access control measures to protect critical control layer content e.g. role-based authorization.

V. SECURE CONTROLLER INTERFACES

There are three potential interfaces to the SDN controller: the D-CPI, A-CPI, and I-CPI for Data-Controller, Application-Controller, and Intermediate-Controller Plane Interface, respectively [18]. To date, the only standardized interface is the D-CPI and in the OF Switch Specification [18], the use of TLS (Transport Layer Security) is recommended but optional. A further interface to the controller is the Graphical User Interface (GUI). A GUI is commonly provided for ease of network management and provides a network topology viewer, network device information and flow table details. Sensitive communication on any of these interfaces should be protected.

A. Secure Control Layer Communication

A notable improvement with the latest SDN controllers is their support for TLS across the D-CPI. Secure communication with SSL/TLS refers to authentication of the parties to a communication (using X.509 certificates) and subsequent encryption of the data between the parties across the communication

interface. Of the 5 controllers studied, ODL and Ryu support SSL/TLS as tested with the OVS public key infrastructure (PKI). ONOS does not provide SSL/TLS support and there is no mention of SSL/TLS in the description of ROSEMARY. In addition to supporting TLS on the D-CPI, SE-Floodlight also supports secure communication across the A-CPI.

However, as highlighted in [14], SSL/TLSv1 based communication is unable to protect against IP based attacks on the control channel. Therefore, additional protection against IP spoofing, TCP Synchronization (SYN), and Denial-of-Service attacks must be provided. It is recommended that controllers support a secure version of TLS (e.g. v1.2) or a TLS equivalent protocol for communication between the application/data layers and the control layer to mitigate tampering with message exchanges. Furthermore, certificate/key materials should be carefully managed to underpin the security of the PKI.

B. GUI/REST API Security

The ability to manipulate the network state via the controller GUI means that access to the GUI should be protected. ODL, ONOS, and Ryu each provide a GUI for interacting with the controller. Apart from ROSEMARY for which no information is provided, each controller also supports REST for communication across the A-CPI.

In the case of ONOS, there is no authentication/authorization required to access the GUI or applied to the REST calls. From inside the network, the IP address of the device hosting the controller is required. ODL provides some security requiring a user-name/password to log in to the controller GUI, *DLux*. The default user-name and password should be changed. Ryu offers a basic topology viewer rather than a full GUI. The topology viewer provides a graphical illustration of the network topology, link status and flow entries. It is not secured. Despite the extensive security enhancements introduced in SE-Floodlight, the web user interface using REST is not access controlled. For secure deployment, this controller interface should be protected to prevent information disclosure.

VI. Controller Security Services

In addition to protecting the control framework and interfaces, security services can be introduced to the controller.

A. IDS/IPS Integration

Defense4All is provided with ODL and links to the controller via REST API. Traffic monitoring is provided by automatically installing traffic counting flows in selected nodes. The controller produces traffic statistics for attack detection, which is determined by significant deviation from a normal traffic baseline. Attack mitigation is provisioned by diverting traffic through a network-attached Attack Mitigation System (AMS). The AMS design is out of scope of Defense4All.

The ease of integration of Snort [19], an open-source intrusion detection engine, with Ryu is demonstrated with the application, $simple\_switch\_snort.py$, provided with the controller source code. This application installs a flow to mirror incoming packets to the snort network interface. A set of custom rules are generated and a packet matching a custom rule generates a Snort alert that generates an event alert in Ryu. The code can be extended for intrusion prevention.

In [7], a self-defending wireless network is demonstrated using SE-Floodlight with the SRI BotHunter and SDN Security Actuator applications. BotHunter monitors traffic to identify communication patterns consistent with coordination-centric malware, BHResponder decides whether the identified asset should be quarantined from the network, and SDN Security Actuator links with SE-Floodlight to implement the quarantine i.e. generate OF rules to redirect suspicious traffic flows.

ODL, Ryu, and SE-Floodlight provide enhanced network security by integrating these security applications.

B. Authentication and Authorization

Authentication, Authorization, and Accounting (AAA) are important aspects of the controller design in order to provide effective access control to users, applications, and resources. Accounting is discussed in Section VI-D.

Although ONOS does not explicitly implement AAA functions, the applications register with *CoreService* for a unique App ID to use ONOS services. Future versions of ONOS may apply the App ID for AAA functions. The AAA project was launched in ODL mid-2014. The identity of users is authenticated, and user access to resources is authorized and recorded. The user authenticates to the controller with a user-name/password combination and receives an access token to access protected resources on the controller. Access to specific resources is determined by the user role and permissions.

ROSEMARY provides a similar AAA system in which applications are authorized to access specific controller resources. For privileged system calls, an application authorization module determines whether the application is authorized by investigating its signed key. In SE-Floodlight, authorization roles are assigned to applications during an application authentication procedure, which involves generation of a runtime credential to uniquely identify the application. The authorization role includes a set of associated permissions. The credential is added to each message produced by the application. Without the credential, the application will not run.

C. Resource Monitoring

With multiple applications and functions running in the control framework, it is necessary to protect against any single application consuming excessive resources.

ROSEMARY has proposed a resource manager to control application resource utilization (CPU, memory, file descriptor). A resource table is used to manage the maximum resources assigned to each application. At the soft limit, an alert is issued while at the hard limit, resource is capped or the application is terminated. The other controllers studied do not currently implement resource management. The benefit of resource monitoring is not only to effectively manage the available resources to support an optimum number of applications but also to monitor anomalous behaviour to identify potentially malicious or buggy applications.

D. Logging/Security Audit Service

Accounting was identified as a key aspect of AAA (Section VI-B). Retaining logs relating to resource access and events provides valuable information to map the sequence of events leading to a system failure/attack.

Both ONOS and ODL report debug information to a Karaf log. With Ryu, a dumper file can be configured to capture required debug information e.g. specific OF events or statistics. The information can be output to a selected log-file with default permissions. A system log manager is built into the ROSEMARY kernel and is used to capture the warnings and alerts generated by the NOS. SE-Floodlight provides a security audit service for tracking security-relevant events at the OF control layer. A set of relevant events is defined in accordance with the Department of Defense Trusted Computer System Evaluation Criteria. Use of the audit API is restricted through the SEK permissions service with audit messages requiring a minimum of SEC authorization role.

VII. Review and Recommendations for future Security Improvements

It is clear from Table II that there is currently no single SDN controller that includes each of the identified features for a secure, robust, and resilient SDN controller. ONOS and ODL focus on the provision of a distributed architecture while ROSEMARY and SE-Floodlight introduce control layer resilience and a security-enforcement kernel, respectively. In addition to the attributes listed in Table II, the following design recommendations are identified:

(1) *Design with Software Security Principles:* Privilege limitation, secure defaults, and sensitive data encryption are elements of secure software design. Static analysis tools should also be used to test the security of the controller design and detect when an inconsistent network state is reached e.g. NICE [20]. (2) *Secure Default Controller Settings:* The SDN controller should be secure throughout the system life-cycle from initialization to recovery. A safe mode boot process (such as outlined in ROSEMARY) should be fundamental to all SDN controller designs. (3) *Application Future-Proofing:* In order to support application transferability across controller platforms, applications should be designed outside the controller and without OF. This requires abstraction of network communication but will support future D-CPI protocols.

VIII. Conclusions and Future Work

SDN controller design has evolved significantly since the first NOX OF controller was developed for centralized SDN control, with improvements in performance, scalability, and reliability. In this paper, a set of attributes of a secure, robust, and resilient SDN controller have been presented. The extent to which current state-of-the-art open-source controllers support these attributes has been discussed. It is promising that all except one of the defined security attributes is supported by one or more controller. The missing feature is the management of multiple application instances for application resilience. This must be a design consideration for future controller developments. With the clear split between high availability controllers and secure/resilient control layers, the next evolution in SDN controller design must be a means to achieve the combined goal of security, robustness, and resilience.

References

[1] 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.

[2] D. Kreutz, F. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[3] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN Security: A Survey," in *IEEE SDN for Future Networks and Services (SDN4FNS)*, 2013, pp. 1–7.

[4] OPENDAYLIGHT, "OpenDaylight: A Linux Foundation Collaborative Project." [Online]. Available: http://www.opendaylight.org

[5] ON.LAB, "ONOS: Open Network Operating System." [Online]. Available: http://onosproject.org/

[6] Nippon Telegraph and Telephone Corporation, "Ryu Network Operating System." [Online]. Available: http://osrg.github.io/ryu/

[7] P. Porras, S. Cheung, M. Fong, K. Skinner, and V. Yegneswaran, "Securing the Software-Defined Network Control Layer," in *Proceedings of the 2015 Network and Distributed System Security Symposium (NDSS)*, February 2015.

[8] 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.

[9] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, and W. Snow, "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.

[10] M. Wagner, "Opendaylight patches 'serious vulnerability'-after 4 months," 17 December 2014. [Online]. Available: http://www.lightreading.com/carrier-sdn/sdn-technology/opendaylight-patches-serious-vulnerability-andndash-after-four-months/d/d-id/712626

[11] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, vol. 54, 2012.

[12] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of SDN/OpenFlow controllers," in *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia*. ACM, 2013, p. 1.

[13] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based comparison and selection of Software Defined Networking (SDN) controllers," in *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*. IEEE, 2014.

[14] M. Liyanage, M. Ylianttila, and A. Gurtov, "Securing the Control Channel of Software-Defined Mobile Networks."

[15] F. Botelho, A. Bessani, F. M. Ramos, and P. Ferreira, "On the design of practical fault-tolerant SDN controllers," in *Proc. of the 3rd European Workshop on Software Defined Networks-EWSDN*, vol. 14, 2014.

[16] B. Chandrasekaran and T. Benson, "Tolerating SDN application failures with LegoSDN," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. ACM, 2014, p. 22.

[17] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, "Towards a secure controller platform for openflow applications," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 171–172.

[18] "ONF Specifications." [Online]. Available: https://www.opennetworking.org/sdn-resources/onf-specifications

[19] "Snort - Open Source Intrusion Prevention System." [Online]. Available: https://www.snort.org

[20] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford, "A NICE way to test OpenFlow applications," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 2012.