SDN Security: A Survey

Sandra Scott-Hayward, Gemma O'Callaghan and Sakir Sezer
Centre for Secure Information Technology (CSIT)
Queen's University Belfast
Belfast, BT3 9DT, Northern Ireland

***ABSTRACT*—The pull of Software-Defined Networking (SDN) is magnetic. There are few in the networking community who have escaped its impact. As the benefits of network visibility and network device programmability are discussed, the question could be asked as to who exactly will benefit? Will it be the network operator or will it, in fact, be the network intruder? As SDN devices and systems hit the market, security in SDN must be raised on the agenda. This paper presents a comprehensive survey of the research relating to security in software-defined networking that has been carried out to date. Both the security enhancements to be derived from using the SDN framework and the security challenges introduced by the framework are discussed. By categorizing the existing work, a set of conclusions and proposals for future research directions are presented.**

I. Introduction

Software-defined networking (SDN) is rapidly moving from vision to reality with a host of SDN-enabled devices in development and production. The combination of separated control and data plane functionality and programmability in the network, which have long been discussed in the research world, have found their commercial application in cloud computing and virtualization technologies.

The advantages of SDN in various scenarios (e.g. the enterprise, the datacenter etc.) and across various backbone networks have already been proven e.g. Google B4 [1]. However, challenges exist for a full-scale carrier network implementation of SDN. A number of these challenges have been presented in [2]. One key area, which is only beginning to receive the attention it deserves, is that of security in SDN.

The SDN architecture can be exploited to enhance network security with the provision of a highly reactive security monitoring, analysis and response system. The central controller is key to this system. Traffic analysis or anomaly-detection methods deployed in the network generate security-related data, which can be regularly transferred to the central controller. Applications can be run at the controller to analyze and correlate this feedback from the complete network. Based on the analysis, new or updated security policy can be propagated across the network in the form of flow rules. This consolidated approach can efficiently speed up the control and containment of network security threats.

However, the same attributes of centralized control and programmability associated with the SDN platform introduce network security challenges. An increased potential for Denial-of-Service (DoS) attacks due to the centralized controller and flow-table limitation in network devices is a prime example. Another issue of concern based on open programmability of the network is trust; both between applications and controllers, and controllers and network devices.

A number of solutions to these SDN security challenges have been proposed in the literature. These range from controller replication schemes through policy conflict resolution to authentication mechanisms. Similarly, a number of proposals have been made to exploit the SDN framework for enhanced network security.

An analysis of the security challenges of SDN is presented in this paper. The individual security issues are categorized according to the SDN layer affected or targeted. The proposed and emerging solutions to these challenges are then discussed and categorized. The requirement for further work to establish a secure and robust SDN is clearly identified from the gap between the issues and the existing research. Without a significant increase in focus on security, it will not be possible for SDN to support the evolving capability associated with, for example, Network Functions Virtualization (NFV) [3].

II. Security Analyses of SDN

The basic properties of a secure communications network are: confidentiality, integrity, availability of information, authentication and non-repudiation [4]. In order to provide a network protected from malicious attack or unintentional damage, security professionals must secure the data, the network assets (e.g. devices) and the communication transactions across the network. The alterations to the network architecture introduced by SDN must be assessed to ensure that network security is sustained.

In an early iteration of what is known today as SDN, Casado et al. [5] specifically considered the security aspects of a separate control and forwarding framework. Their SANE architecture, proposed in 2006, centred on

a logically centralized controller responsible for authentication of hosts and policy enforcement. At the time of its proposal, this was considered to be an extreme approach that would require a radical change to the networking infrastructure and end-hosts, which could be too restrictive for some enterprises.

Ethane [6] extended the work of SANE but used an approach, which required less alteration to the original network. It controlled the network through the use of two components; a centralized controller responsible for enforcing global policy, and ethane switches, which simply forwarded packets based on rules in a flow table. This simplified network control allowed the data and control plane to be separated to allow for more programmability. Although the Ethane architecture gave us a closer look at what SDN and OpenFlow would become, it suffered from a number of drawbacks. One of these is the fact that application traffic could compromise network policy. In today's SDN architecture, applications are used to provide various services, as, for example, with Network Functions Virtualization (NFV). The compromise of applications could potentially breach the entire network.

Considering the specific issues with security in SDN from the perspective of the SDN framework (Fig. 1), we can identify challenges associated with each layer of the framework: application, control and data planes, and on the interfaces between these layers.

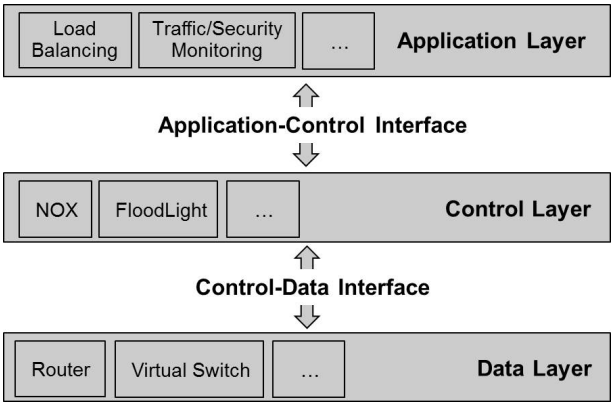

Fig. 1. SDN Functional Architecture illustrating the data, control and application layers and interfaces

A number of security analyses have recently been performed, which have found that the altered elements or relationship between elements in the SDN framework introduce new vulnerabilities, which were not present before SDN. One such paper [7] completes an analysis of the OpenFlow protocol using the STRIDE threat analysis methodology [8]. This paper focuses on the execution of Information Disclosure and DoS attacks, which the author established were possible to successfully execute. Although a number of mitigation techniques are proposed, these techniques are not proven in the work.

The OpenFlow switch specification [9] describes the use of transport layer security (TLS) with mutual authentication between the controllers and their switches. However, the security feature is optional, and the standard of TLS is not specified. The lack of TLS adoption by major vendors and the possibility of DoS attacks are the focus of an OpenFlow vulnerability assessment [10]. The authors found that the lack of TLS use could lead to fraudulent rule insertion and rule modification.

In [11] Kreutz et al. present a high-level analysis of the overall security of SDN. They conclude that due to the nature of the centralized controller and the programmability of the network, new threats are introduced requiring new responses. They propose a number of techniques in order to address the various threats, including replication, diversity and secure components.

Finally, the research network and testbed, ProtoGENI, has also been analyzed [12]. The authors discovered that numerous attacks between users of the testbed along with malicious propagation and flooding attacks to the wider internet were possible when using the ProtoGENI network.

The results of these analyses indicate the range of the security issues associated with the SDN framework. In Table I, a categorization of the SDN security issues is presented. A connection is drawn between the type of issue/attack (e.g. unauthorized access) and the SDN layer/interface affected by the issue/attack.

The control and data layers are identified in Table I as clear targets of attack. This reflects the main distinctions between the traditional network and the SDN; that of the centralized control element and the altered datapath elements to support programmability.

Although this analysis points towards security issues related to the control and data layers, there has been limited research in the field to tackle the challenges. In fact, as detailed in the next section, greater attention has been given to exploring the potential improvements in network security to be derived from the SDN framework.

III. Security Enhancement using SDN

The architecture of a software-defined network introduces potential for innovation in the use of the network. The combination of the global or network-wide view and the network programmability supports a process of harvesting intelligence from existing Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS), for example, followed by analysis and centralized reprogramming of the network. This approach can render the SDN more robust to malicious attack than traditional networks.

TABLE I
CATEGORIZATION OF THE SECURITY ISSUES ASSOCIATED WITH THE SDN FRAMEWORK BY LAYER/INTERFACE AFFECTED

Security Issue/Attack	SDN Layer Affected or Targeted				
	Application Layer	App-Ctl Interface	Control Layer	Ctl-Data Interface	Data Layer
Unauthorized Access e.g.					
Unauthorized Controller Access			✓	✓	✓
Unauthenticated Application	✓	✓	✓		
Data Leakage e.g.					
Flow Rule Discovery (Side Channel Attack on Input Buffer)					✓
Forwarding Policy Discovery (Packet Processing Timing Analysis)					✓
Data Modification e.g.					
Flow Rule Modification to Modify Packets			✓	✓	✓
Malicious Applications e.g.					
Fraudulent Rule Insertion	✓	✓	✓		
Controller Hijacking			✓	✓	✓
Denial of Service e.g.					
Controller-Switch Communication Flood			✓	✓	✓
Switch Flow Table Flooding					✓
Configuration Issues e.g.					
Lack of TLS (or other Authentication Technique) Adoption			✓	✓	✓
Policy Enforcement	✓	✓	✓		

A. The SDN Middle-box

Traditional networks use middle-boxes to provide network security functions. Recently, there has been discussion about the integration of security middle-boxes into SDN exploiting the benefit of programmability to redirect selected network traffic through the middle-box. For example, the Slick architecture [13] proposes a centralized controller, which is responsible for installing and migrating functions onto custom middle-boxes. Applications can then direct the Slick controller to install the necessary functions for routing particular flows based on security requirements.

The FlowTags architecture [14] proposes the use of minimally modified middle-boxes, which interact with a SDN controller through a FlowTags Application Programming Interface (API). FlowTags, consisting of traffic flow information, are embedded in packet headers to provide flow tracking and enable controlled routing of tagged packets. A clear disadvantage of this architecture is the fact that it works with only pre-defined policies and currently does not handle dynamic actions.

The SIMPLE policy enforcement layer [15] is an approach for using SDN to manage middlebox deployments. In contrast to [13], [14], it requires no modifications to SDN capabilities or middle-box functionality, which makes it suitable for legacy systems.

Based on these proposals, it would appear that a simple approach to network security provision would be to introduce an appropriate middle-box and programme the network to direct selected traffic through the middle-box. It is not, however, quite as straightforward as that. The appropriate placement and integration of SDN middle-boxes must be determined along with the performance penalty that can be tolerated when traffic is diverted through an additional link. Such questions have not yet been resolved.

However, as illustrated in Table I, the range of attacks that pose threats to the network is well understood. As such, beyond middle-boxes, a series of solutions have been proposed, which specifically exploit the SDN framework to provide network security solutions.

B. SDN = "Security Defined Networking"?

Attackers use various scanning techniques to discover vulnerable targets in the network. One defense presented to thwart these attacks is the use of random virtual Internet Protocol (IP) addresses using SDN [16]. This technique uses the OpenFlow controller to manage a pool of virtual IP addresses, which are assigned to hosts within the network, hiding the real IP addresses from the outside world. This presents moving target defense, which is a form of adaptive cybersecurity.

Monitoring Systems are essential in protecting the network from attack. In [17], the authors present a Distributed DoS (DDoS) detection method based on several traffic flow features. This system monitors NOX (C++ based OpenFlow Controller) switches at regular intervals and uses Self Organizing Maps to identify abnormal flows. In another approach, OpenSAFE [18] uses its ALARMS policy language to manage the routing of traffic through network monitoring devices. A similar idea focusing on SDN in the cloud was presented by Shin and Gu in [19]. CloudWatcher controls network flows to guarantee that all necessary network packets are inspected by some security devices. This framework automatically detours network packets to be inspected by pre-installed network security devices.

These solutions are based on a centralized network management scheme; however other work encourages the delegation of some control back to network devices and hosts. Resonance, for example, [20], provides dynamic access control enforced by network devices themselves based on higher-level security policies. Naous et al. [21] put forward the ident++ protocol to query end-hosts and users for additional information in order to make forwarding decisions; their argument being that the central controller could become a bottleneck. While retaining the programmability characteristic of SDN, these methods propose to involve the network devices in the control of the network, rather than relying on a single, centralized controller.

One specific form of monitoring system, the IDS, has been the focus of a number of SDN solutions. Skowyra et al. [22] propose a learning IDS, which utilizes the SDN architecture to both detect and respond to network attacks in embedded mobile devices. A hardware-accelerated NIDS (Network IDS) or NIPS (Network IPS) scheme, as described in [23], allows the network administrator to configure string patterns for use by a deep packet inspection (DPI) module. Finally, the value of using SDN to provide intrusion detection in a Home Office/Small Office environment is proposed in [24].

The possibility for improving and simplifying network security by means of the SDN architecture is evident from this body of research. This potential has also been recognised commercially with a range of SDN security products at various stages of development.

IV. Security Challenges with SDN

While security as an advantage of the SDN framework has been recognized, solutions to tackle the challenges of securing the SDN network are fewer in number.

SDNs provide us with the ability to easily program the network and to allow for the creation of dynamic flow policies. It is, in fact, this advantage that may also lead to security vulnerabilities. Within this dynamic environment, it is vital that network security policy is enforced. Model-checking becomes an important step in detecting inconsistencies in policies from multiple applications or installed across multiple devices. Model checking combined with symbolic execution may be used to test OpenFlow applications for correctness [25]. Binary Decision Diagrams can also be used to test for intra-switch misconfigurations within a single flow table [26]. FlowChecker exploits FlowVisor [27], which enables isolation by partitioning the network resources into slices. Son et al. propose Flover [28], which uses assertion sets and modulo theories to verify flow policies, while VeriFlow [29] studies the verification of invariants in real-time. An additional layer, which sits between the SDN controller and the network devices, intercepts flow rules before they reach the network. Although VeriFlow boasts low-latency of the checking process, it cannot handle multiple controllers. In [30], the authors propose the use of language-based security to enable flow-based policy enforcement along with network isolation. This solution is implemented as a NOX application and allows the integration of external authentication sources to provide access control. More recently, Splendid Isolation [31] has been proposed as a means of verifying the isolation of program traffic. This programming model supports the idea of network slices to provide the fundamental security concepts of confidentiality and integrity. There is a clear emphasis from the research community on this issue of policy conflict resolution.

However, proposals to aid in the design of secure SDNs are limited. Fresco [32] is one notable contribution; which presents an OpenFlow Security Application Development Framework incorporating FortNox [33]; a security enforcement kernel. The idea behind FRESCO is to allow the rapid design and development of security specific modules, which can be incorporated as an OpenFlow application. Porras et al. provide a library of reusable modules which can be used for the detection and mitigation of network threats. This system incorporates the FortNox enforcement engine, which handles possible conflicts with rule insertion. If a rule conflict arises as a result of a new OpenFlow rule enabling or disabling a prohibited/allowed existing rule, then the new rule is accepted or rejected depending on the level of security authorization of the author to the existing conflicting rule provider. Although FortNox provides numerous components, which are necessary for enforcing security, the authors feel that much work is still needed to offer a comprehensive suite of applications.

Moving from the design space to implementation, one of the key industry concerns with security in SDN is satisfaction of the audit process. For network compliance and operation, a controlled inventory of network devices is required. This involves knowledge of what devices

TABLE II
CATEGORIZATION OF THE RESEARCH ON SECURITY IN SDN

Research	Security			OpenFlow	SDN Layer/Interface				
Work	Analysis	Enhancement	Solution		App	App-Ctl	Ctl	Ctl-Data	Data
[7], [10], [12]	✓			✓			✓	✓	✓
[11]	✓				✓		✓	✓	✓
[5]							✓	✓	✓
[13], [14], [21]		✓		✓	✓	✓	✓	✓	✓
[15]		✓			✓		✓		✓
[16]		✓		✓			✓	✓	✓
[17], [24]		✓		✓	✓		✓	✓	
[18], [19]		✓		✓	✓	✓	✓	✓	
[20], [22]		✓		✓	✓		✓	✓	✓
[23]		✓			✓				✓
[25]			✓	✓	✓	✓		✓	
[26], [28]–[30], [32]			✓	✓	✓	✓	✓	✓	
[31]			✓			✓	✓		
[33], [34]			✓	✓		✓	✓	✓	
[35]			✓	✓	✓			✓	

are running, how they are bound to the network etc. This directly concerns the potential for virtualization of network elements and functions as supported by the SDN framework. Although there is an unresolved challenge regarding the feasibility of mapping network state across mobile and virtual functions, some related work regarding network verification is worth mentioning. In [34], the authors consider the problem of scalability and security of OpenFlow networks and their use in the cyber-physical space. Verificare allows for specification modeling and verification of network correctness, convergence and mobility-related properties. Hadigol et al. propose the use of a prototype network debugger [35], which could be used to allow SDN developers to reconstruct the chain of events which lead to a bug and identify its root cause.

As identified in Section II, the SDN architecture can be considered as a set of layers and interfaces. The layer/interface affected by some of the SDN-specific security issues was identified in Table I. In a similar manner, the SDN security research work is classified in Table II by the layer/interface, which the analysis, enhancement or solution targets. The results of this categorization are discussed in the next section. It can be noted that SANE [5] is included in Table II for categorization with respect to affected layers/interfaces. However, as a separate architecture, it is not identified as an SDN security enhancement or solution.

V. DISCUSSION

Considering the categorization of research work in Table II, it can be seen that there has been greater focus on exploiting SDN for enhanced network security than on generating solutions to the identifed security issues. The enhancement work has centred on the use of middle-boxes and monitoring systems for security service insertion to dynamically detect and/or prevent suspicious traffic during live network operation.

There is further potential in this area to exploit the dynamic and adaptive capabilities of the SDN framework using methods of moving target defense. The work presented in [16] is one such example where randomizing the virtual IP addresses makes it more difficult for an attacker to breach the network. Without a fixed system to observe and prepare to attack, the strength of the attacker is reduced.

New methods and techniques must be explored to expand on the programmability of the network enabling dynamic adjustments in security monitoring, detection and prevention capabilities.

A minor observation from the content of Table II is that the majority of the work references or implements OpenFlow for the control-data interface. Although any alternative to OpenFlow would have similar attributes, it is worth noting that OpenFlow may not be the only/definitive control-data interface protocol in SDNs. For example, several Internet Engineering Task Force (IETF) groups have defined protocols regarding separation of forwarding and control planes, network configuration and routing. These include IETF ForCES (Forwarding and Control Element Separation), PCE (Path Computation Element), Netconf (Network Configuration), LISP (Locator/ID Separation Protocol)

and I2RS (Interface to the Routing System). In addition, proprietary protocols are being developed by individual companies. The work to identify and correct security-related limitations of the OpenFlow protocol should be considered in the design and development of alternative protocols. This could apply both to the control-data plane interface and also to the higher-level abstractions at the application-control interface, which may present similar concerns.

The most significant element to highlight from the categorization of security-related SDN research is that there is an identifiable disconnect between the security analyses presented to date, which focus on the control-data plane issues, and the solutions to security issues, the majority of which focus on one application-control plane issue; that of policy conflict resolution.

Considering the breadth of potential security issues outlined in Table I, it is clear that a significant increase in effort is required to identify solutions to these challenges.

This requirement has been recognised in the past year in some areas of the networking community. Since the beginning of 2013, various working groups have been established in both the standardization industry and industry research groups. In the Open Networking Foundation (ONF) and the European Telecommunications Standards Institute (ETSI), groups focussed specifically on security in SDN and NFV, respectively, have been launched. In the Internet Research Task Force (IRTF) and the International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), general SDN study groups have been launched in which security in SDN is an identified issue.

One of the recurring themes from these industry working groups is the importance of designing security in from the start. By this, it is meant that while SDN is in the early stages of development, the associated security issues should be identified and resolved. However, SDN-compliant hardware, software and services are already in production and in service. While some of these solutions are, in fact, SDN security products, many others have been developed with little or no consideration of the security implications of a wide area network deployment.

It is, therefore, essential, that techniques, methods and policies to overcome the SDN security challenges are explored and defined to enable robust and reliable wide area SDN deployments. An increased emphasis on this now could avoid a reduction in the performance and capability of future SDNs as a result of retrofit security solutions.

VI. Conclusion

There are two schools of thought on security in software-defined networking. The first is that significant improvements in network security can be achieved by simultaneously exploiting the programmability and the centralized network view introduced by SDN. The second is that these same two SDN attributes expose the network to a range of new attacks. In this article, we have categorized the SDN security challenges and presented a comprehensive review of the research work on security in SDN to date. Our analysis identifies that regardless of your school of thought, there is yet more to be done; more untapped potential and more unresolved challenges. A concerted effort in both directions could yield a truly secure and reliable Software-Defined Network.

References

[1] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, and M. Zhu, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of the ACM SIGCOMM 2013 conference*. ACM, 2013, pp. 3–14.

[2] S. Sezer, S. Scott-Hayward, P. 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," *Communications Magazine, IEEE*, vol. 51, no. 7, 2013.

[3] "Network Functions Virtualization - Introductory White Paper," October, 2012. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[4] C. Douligeris and D. N. Serpanos, *Network security: current status and future directions*. Wiley. com, 2007.

[5] M. Casado, T. Garfinkel, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker, "Sane: A protection architecture for enterprise networks," in *USENIX Security Symposium*, 2006.

[6] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 1–12.

[7] R. Kloeti, "OpenFlow: A Security Analysis," April 2013. [Online]. Available: ftp://yosemite.ee.ethz.ch/pub/students/2012-HS/MA-2012-20_signed.pdf

[8] S. Hernan, S. Lambert, T. Ostwald, and A. Shostack, "Threat modeling-uncover security design flaws using the stride approach," *MSDN Magazine-Louisville*, pp. 68–75, 2006.

[9] "OpenFlow Switch Specification Version 1.3.2," Open Networking Foundation. [Online]. Available: https://www.opennetworking.org

[10] K. Benton, L. J. Camp, and C. Small, "OpenFlow Vulnerability Assessment," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 151–152.

[11] D. Kreutz, F. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 55–60.

[12] D. Li, X. Hong, and J. Bowman, "Evaluation of Security Vulnerabilities by Using ProtoGENI as a Launchpad," in *Global Telecommunications Conference (GLOBECOM 2011)*. IEEE, 2011, pp. 1–6.

[13] B. Anwer, T. Benson, N. Feamster, D. Levin, and J. Rexford, "A Slick Control Plane for Network Middleboxes," *Open Networking Summit*, 2013. [Online]. Available: http://nextstep-esolutions.com/Clients/ONS2.0/pdf/2013/research_track/poster_papers/final/ons2013-final51.pdf

[14] S. Fayazbakhsh, V. Sekar, M. Yu, and J. Mogul, "FlowTags: Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions," in *Proceedings of the second workshop on Hot topics in software defined networks*. ACM, 2013.

[15] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying Middlebox Policy Enforcement Using SDN." ACM SIGCOMM, August 2013.

[16] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: transparent moving target defense using software defined networking," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 127–132.

[17] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *IEEE 35th Conference on Local Computer Networks (LCN)*. IEEE, 2010, pp. 408–415.

[18] J. R. Ballard, I. Rae, and A. Akella, "Extensible and scalable network monitoring using opensafe," *Proc.INM/WREN*, 2010.

[19] S. Shin and G. Gu, "CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)," in *20th IEEE International Conference on Network Protocols (ICNP)*. IEEE, 2012, pp. 1–6.

[20] A. K. Nayak, A. Reimers, N. Feamster, and R. Clark, "Resonance: dynamic access control for enterprise networks," in *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009, pp. 11–18.

[21] J. Naous, R. Stutsman, D. Mazieres, N. McKeown, and N. Zeldovich, "Delegating network security with more information," in *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009, pp. 19–26.

[22] R. Skowyra, S. Bahargam, and A. Bestavros, "Software-Defined IDS for Securing Embedded Mobile Devices," 2013. [Online]. Available: http://www.cs.bu.edu/techreports/pdf/2013-005-software-defined-ids.pdf

[23] A. Goodney, S. Narayan, V. Bhandwalkar, and Y. H. Cho, "Pattern Based Packet Filtering using NetFPGA in DETER Infrastructure." [Online]. Available: http://fif.kr/AsiaNetFPGAws/paper/2-2.pdf

[24] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Recent Advances in Intrusion Detection*. Springer, 2011, pp. 161–180.

[25] 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.

[26] E. Al-Shaer and S. Al-Haj, "FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures," in *Proceedings of the 3rd ACM workshop on Assurable and usable security configuration*. ACM, 2010, pp. 37–44.

[27] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech.Rep*, 2009.

[28] S. Son, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Model Checking Invariant Security Properties in OpenFlow." [Online]. Available: http://faculty.cse.tamu.edu/guofei/paper/Flover-ICC13.pdf

[29] A. Khurshid, W. Zhou, M. Caesar, and P. Godfrey, "VeriFlow: Verifying network-wide invariants in real time," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 467–472, 2012.

[30] T. Hinrichs, N. Gude, M. Casado, J. Mitchell, and S. Shenker, "Expressing and enforcing flow-based network security policies," *University of Chicago, Tech.Rep*, 2008.

[31] C. Schlesinger, A. Story, S. Gutz, N. Foster, and D. Walker, "Splendid isolation: Language-based security for software-defined networks," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 79–84.

[32] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, "FRESCO: Modular composable security services for software-defined networks," in *Proceedings of Network and Distributed Security Symposium*, 2013.

[33] 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.

[34] R. W. Skowyra, A. Lapets, A. Bestavros, and A. Kfoury, "Verifiably-safe software-defined networks for CPS," in *Proceedings of the 2nd ACM international conference on High confidence networked systems*. ACM, 2013, pp. 101–110.

[35] N. Handigol, B. Heller, V. Jeyakumar, D. Mazires, and N. McKeown, "Where is the debugger for my software-defined network?" in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 55–60.