Towards Secure Multi-tenant Virtualized Networks

Nicolae Paladi
SICS Swedish ICT
nicolae@sics.se

Christian Gehrmann
SICS Swedish ICT
chrisg@sics.se

***Abstract*—Network virtualization enables multi-tenancy over physical network infrastructure, with a side-effect of increased network complexity. Software-defined networking (SDN) is a novel network architectural model – one where the control plane is separated from the data plane by a standardized API – which aims to reduce the network management overhead. However, as the SDN model itself is evolving, its application to multi-tenant virtualized networks raises multiple security challenges. In this paper, we present a security analysis of SDN-based multi-tenant virtualized networks: we outline the security assumptions applicable to such networks, define the relevant adversarial model, identify the main attack vectors for such network infrastructure deployments and finally synthesize a set of high-level security requirements for SDN-based multi-tenant virtualized networks. This paper sets the foundation for future design of secure SDN-based multi-tenant virtualized networks.**

***Index Terms*—Security; Software Defined Networks; Network Virtualization; Multi-tenant Virtualized Networks**

I. Introduction

Rapid development of cloud services made a successful case for virtualization, which allows infrastructure providers to multiplex physical resources and provide complete platform and network resources to multiple tenants.

Multi-tenant cloud infrastructure relies on virtualization of both hosts and network infrastructure. In both cases, system virtualization presents a trade-off between portability and tenant isolation on one hand, and virtualization overhead on the other hand. For host virtualization, *virtual machines* (VMs) are a widely used approach to enable multi-tenancy in the Infrastructure-as-a-Service cloud model. Similarly, a variety of approaches exist for network virtualization, operating on different levels. Sherwood [1] described five dimensions that must be *sliced* to enable network virtualization: bandwidth, topology, traffic, device CPU and the forwarding tables (also called *forwarding information base*, FIB). Given the inherently distributed nature of network infrastructure, no single component modification can satisfactorily slice the network across all five dimensions. Multi-tenancy is among the capabilities enabled by virtualization. In the context of infrastructure clouds, we consider the following characteristics of network multi-tenancy:

- A tenant corresponds to a customer using a particular virtual network;
- Tenants may belong to different administrative domains;
- Tenants expect network isolation of their domain;
- Physical resource sharing is fully abstracted, with tenants unaware of other neighbours;
- Tenants may create multiple distinct virtual network instances and topologies.

By enabling network multi-tenancy with strong isolation, network virtualization allows infrastructure providers to multiplex the network infrastructure among network service providers, paving the way for new services and better hardware resource utilization. However, network infrastructure multi-tenancy comes at the cost of increased complexity, leading to higher management costs and new security risks. Software-defined networking (SDN) is a network architectural approach evolved from the "Clean slate" initiative [2], which proposed to decouple the network forwarding functionality from the control and management logic. The initiative aimed to improve network management flexibility based on clear network abstractions: the *management applications*, which expresses the operator goals on a high level; the *network hypervisor*, which implements control program instructions based on a global network view and computes forwarding state for search router/switch; the *network operating system* (NOS), builds the global network view and implements configurations on switches; and finally the *routing and switching equipment*, which forwards packets as instructed. This paved the way for the wide-scale use of commodity hardware for network infrastructure, flexible software implementation of network functionality and new network virtualization abstractions. The SDN architectural approach continues to be a work in progress. Despite a large body of contributions to the SDN architecture ([3], [4], [5], [6], [7], [8]), network operating systems ([9], [10], [11], [12]) and the communication between the data plane and the control plane (also referred to as "southbound API", [13]), SDN continues to evolve, requiring further attention to aspects such as security, scalability and policy enforcement.

A. Contribution

In this paper, we review the security challenges for multi-tenant virtualized network infrastructure based on the SDN architecture. We briefly describe the application of the SDN approach to the multi-tenant virtualized network infrastructure. We introduce an adversarial model suitable for multi-tenant virtualized network infrastructure using the SDN architecture and identify a set of relevant attack vectors. Finally, we present a list of security requirements towards SDN-based multi-tenant virtualized network infrastructures.

B. Organization

The paper is organized as follows: in Section II we present the related work; next, we introduce the adversarial model for SDN-based multi-tenant virtualized network infrastructures in Section III, followed by a description of the relevant attack vectors in Section IV. We continue by listing the requirements for SDN-based multi-tenant virtualized network infrastructures in Section V and conclude in Section VI.

II. Related Work

In this section, we provide an overview of related work on secure SDN architectures.

In [4], the authors presented Ethane (Figure 1), an enterprise network architecture which allows network managers to control the network using a unified interface. Reflecting the identified enterprise network characteristics, the Ethane network consists of commodity switches and one or multiple controllers. The former are responsible for maintaining the FIB and contain a local switch manager to communicate with the controller. The latter handles host registration and authentication, tracks network bindings, verifies permissions and grants access, as well as enforces resource limits on the managed flows. In addition, the authors described a high-level policy definition language for network management policies.

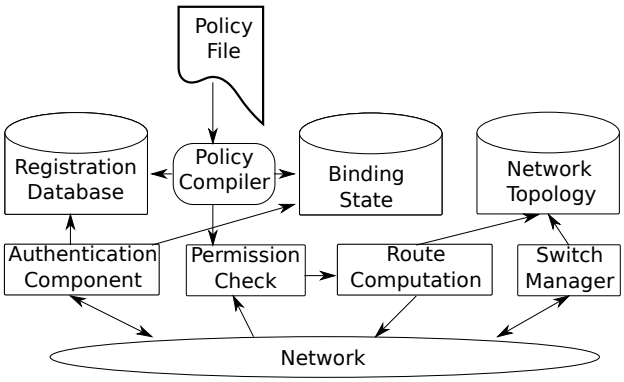

Fig. 1. The main components in Ethane [4]

NOX was introduced in [9], as a network operating system which presents network management programs with a *centralized* programming model and a global view of the system state. This allows network management programs to rely on simpler graph processing algorithms to compute the shortest paths and to operate with higher-level abstractions, such as users and host names, rather than MAC and IP addresses. NOX consists of several distinct *controller* processes operating on a global network view built by NOX based on the process communication with the data plane. The global view is based on switch topology, user location, connected network components (e.g. hosts and middleboxes), as well as bindings between the names and addresses. Controller processes use the global view to make management decisions, which are later implemented on the routing and switching equipment over the OpenFlow API [13].

In [11], the authors presented FortNOX, a software extension for role-based authorization and security constraints enforcement for the NOX OpenFlow Controller. FortNOX detects rule conflicts, i.e. situations when candidate OpenFlow rules modify network flows specified by existing rules, and takes appropriate actions, depending on the authorization of the rule requestor. Role-based authentication is used to determine the security authorization of each rule producer, enforcing the principle of least privilege to ensure integrity of the mediation process. FortNOX consists of four components: a role-based source authentication module to validate signatures for each flow rule insertion request; a conflict analyzer to evaluate each new flow rule against existing active flow rules; a state table manager to track the current flow rules; and finally a flow rule time-out callback interface to update the aggregate flow table upon rule expiration. For conflict resolution, FortNOX converts all flow rules into 'Alias Reduced Rules', allowing to perform a rule set conflict evaluation. FortNOX includes a security directive translator ifor the *block, deny, allow, redirect, quarantine, undo, constrain,* and *info* directives; such directives are used for high-level threat mitigation which are in turn translated to flow rules to handle suspicious traffic. FortNOX has been implemented and evaluated as an extension to NOX and found to introduce an average overhead of less than 7 ms for evaluating a candidate flow rule against 1000 existing flow rules.

In [7] the authors presented 'Fresco', an OpenFlow security application development framework, which facilitates rapid prototyping of composable detection and mitigation modules. Such modules represent elementary building blocks of Fresco and contain the following interfaces: *input, output, event, parameter,* and *action*. The main functions provided by Fresco are script to module translation, database management, event management and instance execution. Implementation of policies defined by composable applications is ensured by the Security Enforcement Kernel built into the fabric of the network operating system. Furthermore, the paper contains a description of a collection of Fresco modules, several composed security applications and a performance evaluation of Fresco. Fresco can be integrated as a security extension module into other network operating systems – such as NOX.

Finally, in "Rosemary: A Robust, Secure, and High-Performance Network Operating System" [12], the authors present a network operating system focusing on network resilience in the presence of faulty or malicious applications by creating sandboxed environments for network applications. Sandboxing (also called micro-NOS) is achieved by launching each application in a separate process context with access to all of the libraries that the application requires. Each Micro-NOS also contains a resource monitor to supervise the applications and operates within the permission structure of Rosemary network operating system. In turn, the Rosemary network operating system is an application running on a commodity Linux distribution. The isolation offered by the micro-NOS allows to improve robustness, such that faulty or malicious applications are prevented from crashing the entire

network operating system. Furthermore, the paper also aims to address security aspects in order to prevent malicious network applications from accessing internal data structures of other network applications. This is achieved by implementing an AppZone sandbox, where privileged system calls made by a network application are interposed and verified by the sandbox framework. To avoid the declared '20-30%' performance overhead, the authors recommend two optimisations called 'request pipelining' and 'trusted execution'. The latter in essence removes the sandbox isolation, allowing the application to run as a kernel process; however, this makes the security advantages of Rosemary less evident. Finally, the authors present the performance evaluation results, which show that the Rosemary network operating system performs roughly on par with the NOX [9] approach on a 1G link and can perform on par with NOX on a 10G link with the 'trusted execution' optimisation in place. This contribution highlights the need for improved security in the architectures of the proposed network operating system. However, one major drawback of the proposed approach is that it ignores distribution aspects, despite significant progresses in distributed network operating system design [10] and the demonstrated need for physical distribution of the control plane [14].

Kreutz et al presented a list of seven attack vectors identified in SDNs [15]: **a.** Forged or faked traffic flows; **b.** Attacks on vulnerabilities in switches; **c.** Attacks on control plane communications; **d.** Attacks on and vulnerabilities in network controllers; **e.** Lack of mechanisms to ensure trust between the controller and management applications; **f.** Attacks on and vulnerabilities in administrative stations; **g.** Lack of trusted resources for forensics and remediation. However, only part of the above attack vectors are exclusively relevant to SDN networks. In this paper, we focus on attack vectors specific for SDN-based multi-tenant virtualized network architectures.

The SDN network architectural model abstracts the complexity of network virtualization; however, in adopting it we must revisit the set of network virtualization mechanisms, identify the most relevant attack vectors, define a relevant adversarial model and outline a set of security requirements towards SDN-based multi-tenant virtualized networks.

III. System and Adversarial Model

In this section, we introduce the SDN system model followed by an example scenario. We next define an adversarial model for SDN-based multi-tenant virtualized networks.

A. SDN system model

A conceptual model of the SDN architecture is depicted in Figure 2, and described below based on the SDN architectural model presented in [8].

- The *data plane* contains both hardware and software routing equipment. This component implements the routing policies that satisfy the goals of the network administrator. It lacks decision logic and is optimized for forwarding speed. Packets that do not match any policy are either discarded or communicated to the control plane through the southbound API.
- *Southbound API* is a vendor-agnostic set of instructions implemented by the routing equipment on the data plane. It allows bi-directional communication between the data and the control planes.
- *Control plane* is a logically distributed abstraction layer that transforms high-level network operator goals into discrete routing policies based on a global network view. It contains a distributed *network operating system*, which builds and maintains the global network view as well as communicates with the equipment on the data plane. The control plane also includes the *network hypervisor*, which multiplexes the available network resources among multiple users with distinct virtual network topologies.
- *Management applications* are used by network administrators to express their network configuration goals using a set of high-level comments. They could also include software-based network management components such as firewalls, intrusion detection systems, traffic shapers, etc.

The above SDN system model will be used as a basis for the adversarial model and threat analysis in the subsequent sections.

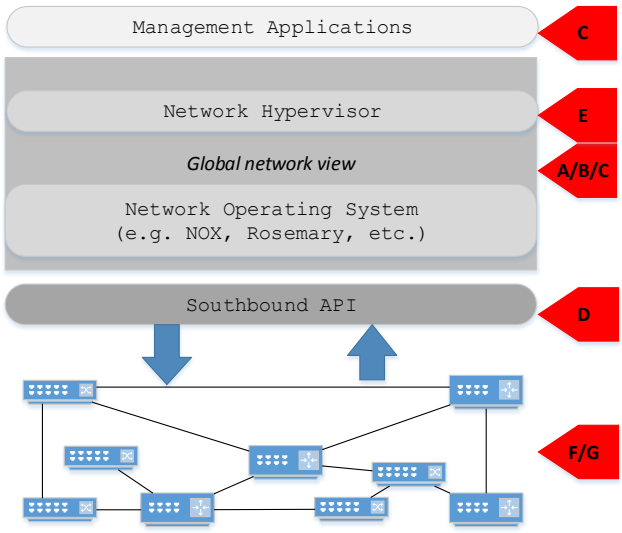

Fig. 2. SDN system model. Letters mark attack vectors, presented in Section IV.

B. SDN Multi-tenant Network Scenario

As an example scenario, consider the cloud network infrastructure model, where tenants purchase virtual network resources (e.g. bandwidth, switching capabilities) from network infrastructure providers. Tenants are represented by the *Virtual Topologies* layer in Figure 2. The parameters of the virtual network resources agreed upon by the network infrastructure provider and tenants are defined by a Quality of Service (QoS) agreement.

In a shared, multi-tenant virtual network environment, the tenants only have a view of the network infrastructure limited to their administrative domain and are not aware about the presence of other tenants. Furthermore, tenants do not have direct access to the configuration of the underlying data plane infrastructure and instead administer their own network domain through routing policies, which are then interposed by the network hypervisor and communicated through the southbound API to the data plane. Enabled by the capabilities of the SDN architectural model, tenants launch their own network management applications (as defined in Section III-A). Such network management applications can be made available both the network infrastructure provider and by third party application providers.

C. Adversarial model assumptions

We present our assumptions regarding SDN-based multi-tenant virtualized networks in the presence of an adversary.

Assumption of hardware integrity: Recent media revelations have raised the issue of hardware tampering en route to deployment sites [16], [17]. We assume that the cloud provider has taken necessary technical and non-technical measures to prevent such hardware tampering.

Assumption of physical security: We assume physical security of the data centres where the network infrastructure is deployed. This assumption holds both when the network infrastructure provider owns and manages the respective data center (as in the case of Amazon Web Services, Google Compute Engine, Microsoft Azure, etc.) and when the network infrastructure provider utilizes the capacity of a data center operated by a third party (e.g. CloudSigma), since physical security can be observed, enforced and verified through known best practices by third-party organizations. This assumption is important for building higher-level hardware and software security guarantees for the components of the network infrastructure.

Assumption of cryptographic security: We assume symmetric and public-key encryption schemes are semantically secure and that the adversary cannot obtain the plain text of encrypted messages. We also assume the signature scheme is unforgeable, and that the message authentication code algorithm correctly verifies message integrity and authenticity. Finally, we assume that the adversary, with a high probability, cannot predict the output of a pseudorandom function.

D. Adversary capabilities

We next describe specific capabilities for adversaries (denoted by $\mathcal{ADV}$). We adopt the Yao-Dolev threat model [18], such that the adversary can overhear, intercept, and synthesize any message and is only limited by the constraints of the employed cryptographic methods. Furthermore, we assume that the adversary can analyse the traffic patterns in the network through passive attacks and may disrupt or degrade network connectivity to achieve its goals, such as e.g. force the sender and the receiver to choose a less secure form of communication. While we prioritise adversaries aiming to compromise the confidentiality and integrity of data in network infrastructure deployments, we also aim to limit the capabilities of attackers to mount Denial-of-Service attacks to disrupt connectivity. We denote this as $\mathcal{ADV}$ A.

We define two additional complementary adversarial types. Acting as a tenant (e.g. through impersonation), the adversary obtains new capabilities in addition to the ones described above; we define this as $\mathcal{ADV}$ B. In this case the adversary is able to perform the following actions using valid network tenant credentials[1]:

1) Send valid tenant packages with an arbitrary content and frequency to the components it can reach;
2) Attempt to impersonate other tenants;
3) Install arbitrary management applications and issue arbitrary policies within its network domain;
4) Use the cryptographic material at its disposal to attempt to decrypt intercepted network traffic that is sent and received by other tenants.

Furthermore, the adversary may manage to take over one of the SDN controller components or some control of the network operating system. We denote this as $\mathcal{ADV}$ C; it will be able to perform the following actions:

1) Affect the network communication of the SDN-based infrastructure by sending valid packets with an arbitrary content and frequency to all reachable network components;
2) Attempt to impersonate network infrastructure components;
3) Issue malicious policies aiming to either monitor, distort or disrupt network traffic;
4) Use the cryptographic material at its disposal to attempt to decrypt intercepted network traffic that is sent and received by other network infrastructure components.

IV. Attack vectors

We review the attack vectors relevant for SDN-based multi-tenant virtualized networks considering the adversarial models presented in Section III.

A. Vulnerabilities in the control plane

Along with ease of network administration, a central control plane introduces a primary attack target for an adversary motivated to take control of the network. Taking over the control plane component in the SDN architecture allows the adversary to obtain full control of the network communication, different from traditional networks where communication control is distributed throughout various network components. Possible solutions include splitting the controller into several domains or distributing the control plane over several hosts, such that issued policies are verified on a different component before deployment.

[1]Related adversary capabilities are defined in: https://tools.ietf.org/html/draft-ietf-nvo3-security-requirements-04

B. Attacks on control plane communications

To manipulate network policies, the adversary may attempt to spoof the control plane communication (both among the components of a distributed controller and between management applications, controller and data plane). Possible solutions include enforcing authenticated and encrypted communication between all of the control plane components, as well as secure enrolment mechanism for management applications and data plane devices.

C. Lack of a trust chain between the management applications and the data plane

While the effort on defining the SDN architecture is still in progress, it is clear that management applications belong to a different security domain than the network operating system, and can be launched by malicious administrators or issue conflicting policies. Both *detecting* and *preventing* malicious policy deviations is challenging: a tenant can only observe the traffic after a change has been applied, but can not obtain and examine snapshots of the data plane FIB; similarly, there is no mechanism to establish a trust chain between tenant commands and entries in the FIB. Possible solutions can be adapted from the ones employed – with varying success – on platform operating systems: verification of code origin and information flow control; however, such mechanisms do not satisfy malicious policy detection requirements.

D. Attacks on policies and rules in programmable networks

Even if the integrity of policies remains intact, the adversary may issue malicious policies that modify or disable the effect of legitimate policies already in place (specifically in the scenario with such network management applications implement functionality of network middleboxes). This type of attack is difficult to detect and prevent, since the malicious policies might be indistinguishable from legitimate ones up to the point when the combined policy is deployed (furthermore, it requires a robust definition of a "malicious policy"). Possible solutions are to establish policy hierarchies and perform policy integration verification against some pre-determined invariants prior to deployment, to ensure that the resulting modifications remain within the basic policy framework. As policy updates may occur interactively in response to changing network patterns, both static analysis of policies and a pre-deployment simulation may be required.

E. Resource limit violations

A malicious tenant may deploy network management applications that exploit vulnerabilities in network service isolation in order to gain network resources beyond the allocated quota defined in the QoS agreement. Possible solutions include adding network operating system capabilities for fine-grained monitoring of management applications to prevent resource overallocation. This in turn requires a well-defined network resource model based on clear definitions of network resources and their respective capacities.

F. Attacks on virtual switches and network gateways

As pointed out in Section III-D, an adversary that controls a virtual network infrastructure component (such as a virtual switch) can attempt to impersonate other virtual network infrastructure components, spoof traffic and negatively affect tenant isolation. Possible solutions include integrity verification of virtual network infrastructure components and protecting the cryptographic secrets necessary for network access using a hardware root of trust.

G. Weak bandwidth isolation as attack vehicle

One of the consequences of NIC virtualization is a weakening of QoS guarantees, since most NIC virtualization implementations do not support guaranteed bandwidth [19]. While this does not directly affect data integrity and confidentiality, manipulating bandwidth allocation between tenants sharing a resource can be used in order to force a policy change (e.g. trigger a more permissive policy that is activated when the available bandwidth falls below a certain threshold). Possible solutions include widespread proliferation of bandwidth isolation techniques such as described in [20], as well as including the effects of bandwidth changes into network policy security testing.

V. Security requirements

In this section, we outline a set of initial security requirements for SDN-based multi-tenant virtualized networks, based on the adversarial model described in Section III and the attack vectors in Section IV:

1) **IV-A:** The SDN control plane must implement an access control model that limits the effects that vulnerabilities in controllers can have on tenant domains. This can prevent an adversary from simultaneously gaining control over the functionality of the SDN controller component at all privilege levels and in all roles.
2) **IV-A:** A dedicated entity must verify the policies to be implemented by the SDN control plane before deployment.
3) **IV-B:** All communication between control plane components must the authenticated, and a secure enrolment mechanism for management applications and data plane devices must be in place.
4) **IV-C:** A mechanism must be in place to offer traceability and non-repudiation for all configuration commands and policies issued by network management applications.
5) **IV-D:** A mechanism must be in place to enforce strong network policy isolation, such that the effects of policies in a certain tenant domain have no effect on other domains. Furthermore, the infrastructure provider must be able to enforce strict boundaries on the effects of policies within tenant domains.
6) **IV-D:** New network management policies must run through an integration verification engine prior to deployment, to minimize or exclude the effect of malicious policies on the network configuration.

7) **IV-E:** A mechanism must be in place to ensure that network management applications do not allocate resources beyond the assigned quota. To do this, the NOS may apply advanced policing mechanisms – e.g. based on existing extensions, such as in [11] – that keep fine-grained tracking of management applications resource utilization and prevent them from making over-allocations.
8) **IV-F:** Integrity of virtual network components must be verified prior to deployment and the cryptographic material required for their network access must be protected with a hardware root of trust.
9) **IV-G:** Policy-based routing decisions must not be affected by vulnerabilities in bandwith isolation between tenants. To clarify, consider a network setup with two types of paths: low-bandwidth, low-cost, low-security permanent paths (*type-A* paths) and high-bandwidth, high-cost, high-security switched paths (*type-B* paths). Consider further that a legitimate tenant has configured a policy to distribute different types of traffic (low-value and high-value traffic) among the *type-A* and *type-B* paths respectively. An adversary capable of modifying the bandwidth allocated to the paths of the legitimate tenant should not succeed in redirecting high-value traffic through *type-A* paths.
10) **IV-G:** Software and hardware network components must offer equally strong bandwidth isolation properties. In the current networks, the data plane components include both software switches and routers deployed on commodity platforms and specialized hardware equipment implemented using application-specific integrated circuits. As pointed out in [19], software-based data plane components lack many of the features currently implemented in specialized hardware switches and routers. Strong bandwidth isolation is one of the features which must be improved in the software implementations.

We plan to address in our forthcoming work the above security requirements towards SDN-based multi-tenant virtualized networks.

VI. Conclusion

Integration of large-scale multi-tenant virtualized network infrastructure with the SDN architectural model presents a set of unsolved security challenges. In this paper, we perform a high-level analysis of SDN-based multi-tenant virtualized networks. We present three security assumptions about such networks, that are necessary for defining a secure virtualized network infrastructure. Based on these assumptions, we define the relevant adversarial model and identify the main attack vectors for such network infrastructure deployments. Finally, based on the defined adversarial model and resulting attack vectors, we outlined a set of high-level security requirements for SDN-based multi-tenant virtualized networks. The adversarial model, attack vector analysis and high-level security requirements defined in this paper serve as an initial input towards future design work of secure and trusted SDN-based multi-tenant virtualized networks.

References

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

[2] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4D approach to network control and management," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 41–54, 2005.

[3] 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*, 2006.

[4] 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, pp. 1–12, ACM, 2007.

[5] 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*, pp. 11–18, ACM, 2009.

[6] H. Song, "Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 127–132, ACM, 2013.

[7] 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.

[8] M. Casado, N. Foster, and A. Guha, "Abstractions for software-defined networks," *Communications of the ACM*, vol. 57, no. 10, pp. 86–95, 2014.

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

[10] 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 *OSDI*, vol. 10, pp. 1–6, 2010.

[11] 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*, pp. 121–126, ACM, 2012.

[12] 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*, pp. 78–89, ACM, 2014.

[13] 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.

[14] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, *et al.*, "B4: Experience with a globally-deployed software defined WAN," in *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 3–14, ACM, 2013.

[15] 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*, pp. 55–60, ACM, 2013.

[16] G. Greenwald, "How the NSA tampers with US-made Internet routers," *The Guardian*, May 2014.

[17] S. Goldberg, "Why is it taking so long to secure internet routing?," *Communications of the ACM*, vol. 57, no. 10, pp. 56–63, 2014.

[18] D. Dolev and A. C. Yao, "On the security of public key protocols," *Information Theory, IEEE Transactions on*, vol. 29, no. 2, pp. 198–208, 1983.

[19] A. Wang, M. Iyer, R. Dutta, G. N. Rouskas, and I. Baldine, "Network virtualization: Technologies, perspectives, and frontiers," *Lightwave Technology, Journal of*, vol. 31, no. 4, pp. 523–537, 2013.

[20] S. Tripathi, N. Droux, T. Srinivasan, K. Belgaied, and V. Iyer, "Crossbow: A Vertically Integrated QoS Stack," in *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, WREN '09, (New York, NY, USA), pp. 45–54, ACM, 2009.