Towards SDN Enabled Network Control Delegation in Clouds

Muhammad Salman Malik, Mirko Montanari, Jun Ho Huh, Rakesh B. Bobba, Roy H. Campbell
University of Illinois, Urbana-Champaign
Email: {mmalik10, mmontan2, jhhuh, rbobba, rhc}@illinois.edu

***Abstract*—In today's IaaS clouds users only get a logical view of the underlying network and have limited control. Delegating more control to end users would be beneficial but would also raise security concerns for the provider. Emerging Software Defined Networking (SDN) technologies have the capabilities to facilitate delegation of network controls and provide some level of network abstractions to end users. However, any delegation solution should try to balance the level of controls delegated to end users with the security constraints of the provider. In this paper, we propose a SDN-based framework to facilitate delegation of some network controls to end users, providing the means to monitor and configure their own slices of the underlying networks. Using two instantiations of this framework, we illustrate the tradeoffs between security and the level of network abstractions provided to end users.**

***Index Terms*—SDN; Cloud; Control Delegation;**

I. Introduction

Cloud computing is gaining popularity, with startups as well as other well established organizations heading towards cloud infrastructures to benefit from its elasticity and also to leverage the lucrative pay-as-you-go model which can lead to considerable cost savings for the cloud users. Infrastructure-as-a-service (IaaS) cloud model, in particular, allows users to build and configure their own virtual computing infrastructure by renting computing platform resources in the form of virtual machines [1]. Users usually get full administrative access to those rented virtual machines, allowing them to install any piece of software they want and configure them to meet their needs. However, the administrative flexibility does not extend to configuring and monitoring the network flows. Users only get a logical view of the underlying network and have very limited control – there are obvious security reasons as to why users only get limited or get no control of the network infrastructure, but there are also technological barriers that are equally responsible for this. Nevertheless, adding more flexibility and transparency in configuring network flows can lead to a more trustable IaaS clouds, and help attract users who are skeptical about cloud providers' SLAs.

Fortunately, with the recent advent of Software Defined Networking (SDN), this goal does not seem too far away from reality. The flexibility provided through SDN has allowed its users to efficiently route their flows in networks [2] and conveniently build security applications on top of their networks [3]. This paper discusses how SDN can be applied in the context of cloud computing to expose the underlying networking infrastructure to the cloud users while balancing the security needs of the infrastructure provider. In particular, our discussions try to understand how existing SDN technologies (e.g., OpenFlow [4], FlowVisor [5], ADVisor [6]) can be utilized to provide adequate level of network path abstractions, while addressing the security concerns of the cloud provider.

It is not hard to imagine scenarios where security-aware users may want to have some *isolation guarantees* on their network flows. For example, they might have requirements for not allowing any other organizations, or perhaps a different group within the same organization, to send to or receive packets from certain parts of their network. Or, they might want their packets to only propagate through certain switch paths, and ensure that they all go through integral security devices (e.g., a firewall or an IDS sensor) before reaching certain destinations. On top of those security needs, those who are running performance-critical jobs might want to ensure that their packets are flowing through optimal paths, meeting some strict response time requirements.

Emerging SDN technologies can enable the implementation of the aforementioned requirements, facilitating a more transparent network flow controls in IaaS clouds, but how would this paradigm-shift affect security of the cloud infrastructure? Our discussions keep two security properties in mind:

- **Non-interference:** cloud users should not be allowed to take control the traffic that does not belong to them;
- **Non-inference:** it should be hard or infeasible for cloud users to map the exact underlying network topology by collecting and reconciling network information over time;

Our aim is not to cover the security of network control delegation mechanism exhaustively, but to discuss the tradeoffs between the two aforementioned security properties that need to be considered when providing more network control to users. Further, it is not our intention to expose a completely virtualized network topology to users, mechanisms for doing so are already available. For example, Amazon EC2 provides a router to which all the VMs of a given user are connected to. Rather, using the network configuring and monitoring capabilities of SDN, we aim to provide a more transparent access to the underlying physical resources, and allow users to better monitor and schedule their network flows (using `multipaths` for example).

The rest of the paper is structured as follows. In Section II we provide necessary background on SDN technologies, OpenFlow, FlowVisor, and ADVisor. We then propose a SDN-based framework, in Section III, that would facilitate flexible

978-1-4799-0181-4/13/$31.00 ©2013 IEEE

network flow controls and monitoring in IaaS clouds. Through two example instantiations of the framework (using FlowVisor and ADVisor respectively) in Section IV, we discuss some of the tradeoffs between security properties and the level of control provided to users. Related work is discussed in Section V. Finally, our conclusions and future work are covered in Section VI.

II. Background

To delegate some control over the network to cloud users, we leverage existing tools to "slice" the network such that each user gets a partial view of the network. Before we delve into the details of our approach, a brief description of these concepts and tools is in order.

A. Software Defined Networks

Software defined networking is a relatively new but revolutionary concept of managing the network. The main idea is to decouple the control logic from the forwarding logic in such a manner that the network can be configured and monitored from a central location (often called controller). Unlike with traditional networks, now network operators can see a global view of the topology at the controller and can apply various data forwarding rules directly from this centralized point. This is a great improvement over the non-SDN networking as it frees the network operators from the burden of configuring each switch separately, *e.g.,* manually configuring switches for firewalls is burdensome and error prone leading to conflicting rules. Note that the communication between controller and the forwarding logic in the switches on the data path is done via a vendor agnostic API. We discuss one such API next.

B. OpenFlow

OpenFlow [4] is an API used in SDNs to control the switches remotely through a centralized controller. This decoupling of the control and data plane provides the network operators with greater flexibility as opposed to the conventional non-SDN routing and management. Every packet for which switches don't know the forwarding rules are sent to the controller and then the controller decides what to do with such packets based on rules or logic specified at the controller. Once a decision has been made by the controller, it pushes *actions* into the switches so that any subsequent packet that matches this flow entry is dealt with accordingly.

C. FlowVisor

FlowVisor [5] is a tool developed for SDNs to slice the physical switches among its users. This slicing of switches can either be on per-port basis of each switch or at a higher level (e.g., data link or transport or network layer level). Network admin can define *flowspaces* (which define the set of packets) for each controller. When a new packet is seen by a switch, it is sent to FlowVisor and then FlowVisor is responsible for directing the packet to its appropriate controller. Since packet rewriting takes place both to and from the controller and switches, FlowVisor acts as a transparent proxy between the controller and switches. Further, it also ensures that a given controller can only see, and hence control, packets that belong to its own flowspace.

D. ADVisor

Advanced FlowVisor (Advisor [6]) is a by-product of OFELIA project from Europe. Basic premise of ADVisor is that although FlowVisor provides slices of the network to its users, the coupling of slices with the underlying topology is stronger than necessary. Particularly, there is no provision in FlowVisor to provide a proper subset of switches in the path that connects two hosts. ADVisor extends FlowVisor to provide users with such an abstraction, i.e., it can provide a proper subset of a path of switches between hosts as a controllable slice to its users.

III. SDN-based Network Control Delegation Framework

This section proposes a SDN-based framework for delegating some network controls to end users while balancing level of delegation with the security concerns of the infrastructure provider. To increase the control that users can have on the underlying network infrastructure, our framework provides the ability for users to deploy their own SDN controllers in the cloud. To provide a concrete example, our framework is described in the context of OpenFlow controllers.

OpenFlow controllers control the network flows by configuring forwarding tables on switches and routers that are redirecting the traffic. A controller has a complete view of the communication flows between the devices in the system. Programs running on it implement complex decisions such as dynamic access control rules, selection of best routes between datacenters [7], and isolation between cloud users. Such control decisions are implemented through OpenFlow messages exchanged between switches and the controller. The level of control provided to cloud users depends on the ability of the cloud user to send and receive OpenFlow messages to an appropriate set of switches.

In any cloud environment, providing a direct unmediated access to the underlying physical infrastructure (e.g., a datacenter) to each end user is undesirable for several reasons:

- malicious or faulty controllers could subvert network communications by violating access control rules or by making communications impossible;
- sensitive information about the physical infrastructure could be revealed, allowing competitors to, for example, infer the size and structure of datacenter or discover security and cost-reduction practices used by the cloud provider;
- making changes to the underlying infrastructure would become much more difficult as all users would have to get involved in reviewing and approving changes to ensure that their controllers can handle the new topology.

For those reasons, our framework exposes only a logical view of the switches present in the system. The degree to which such a logical view matches the physical view provides

a set of tradeoffs between security and flexibility and control given to end users.

User Views: In our framework, given a network with a set of physical switches P, each user receives a logical view of the network V_i. Such a view V_i is composed by removing knowledge about a subset of physical switches, and substituting the remaining ones with logical switches that only give the user partial control over flows through each physical switch. The user writes an Openflow controller interacting with such logical switches, while the cloud provider translates such messages into operations that affect the physical entities.

Using our framework, we can identify several properties that logical views provide. We separate such properties into two groups: *local properties* that depend strictly on the mapping between a single logical network and the physical network, and *global properties* that depend on the interactions between the different views provided to users.

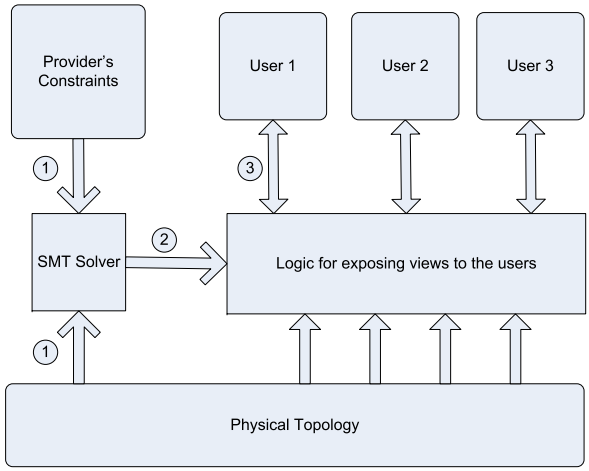

Fig. 1. Conceptual Framework For Delegating Network Control

Local properties: Local properties are properties defined over the view provided to users. We outline a few example of local properties of views. We say that a view satisfies a property of **connectivity** if it exposes to the user a set of switches allowing a controller to set up flows among all its virtual machines. Additionally, we say that a view has a **geographic consistency** property if the view maps physical networks in different regions with networks controlled by different logical switches. In such a view, the location of different virtual machines and their long-distance interconnections are accessible to the user's controller that, on the basis of such information, can optimize network communication.

Similarly, we define a property of **datacenter consistency** that exposes such information at the level of data center. Users with access to a view with datacenter consistency property have knowledge of which communications go across datacenters and can set up flows accordingly. More complex views map redundancy property: a **multi-path** view should allow for users to map at least two network paths between any two virtual machines under their control.

Global properties: Global properties on the other hand are defined over two or more views. For example, we say that two views are **non-interfering** if any action defined by a user in one view cannot affect the other view. Similarly, we say that two views are **fault isolated** if any fault affecting a view cannot affect the other.

Security Properties: The two security properties defined in Section I can be expressed through a combination of global and local properties. For example, we can represent the requirement of limiting the exposure of the cloud provider's physical infrastructure to users through the definition of a global property and a set of local properties. First, we define a global property specifying that the union of views that the cloud users see be a proper subset of the entire network. Second, we define a local property for each view specifying that the subset of switches that are exposed to each user are maximal among all the subset of switches that can be used to provide connectivity among that user's nodes. The first objective relates directly with protecting information about the cloud provider's network, whereas the second objective captures the tradeoff between flexibility that can be provided to a user and the risk of exposing the network to the users. Similarly, the non-interference property can be expressed as a global property that specifies mutual exclusion between user views.

Creating User Views: Given knowledge of the physical infrastructure and the requirement of the users, a cloud provider computes a set of views to provide as follows. Let $\{s_1, s_2, ..., s_N\}$ be the sets of switches that can provide full connectivity between machines under the control of users $\{u_1, u_2, ..., u_N\}$ respectively, then we need to find a subset,s'_i, corresponding to every s_i such that the union of each of these subsets is strictly less than the total number of switches in the cloud provider's network, S. More specifically, if we let boolean variables n_{ij} represent whether i^{th} switch in set s_j lies in s'_j or not, then our network should satisfy the following constraint: $\sum_{i=1}^{N} \sum_{j=1}^{|s_j|} n_{ij} < S$. We can express the given constraint as Satisfiability Modulo Theories (SMT) formula, and use a SMT solver to determine which switches can be exposed to which users without violating the constraints. Figure 1 shows our conceptual framework for delegating network control under certain constraints.

IV. Framework Instantiations and Tradeoff Discussion

In this section, we demonstrate two possible instantiations of the framework using FlowVisor and ADVisor. Based on the two instantiations, we discuss the kind of security tradeoffs that such systems would face while providing different levels

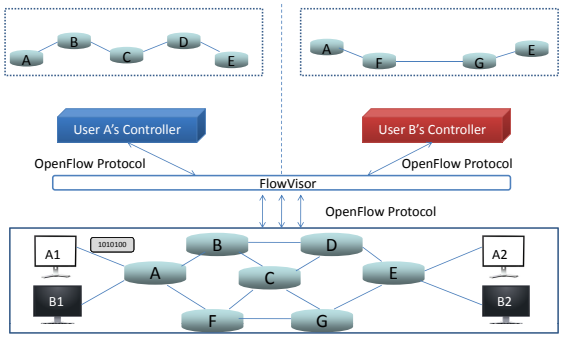

Fig. 2. Architecture for network control delegation using FlowVisor

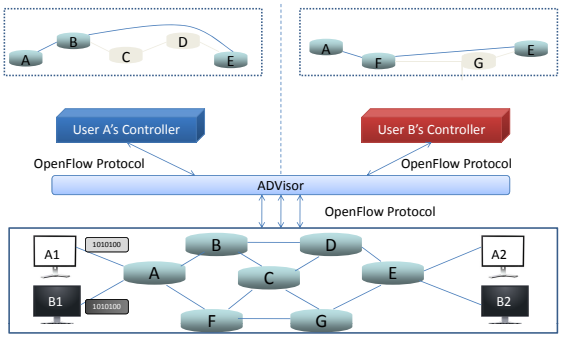

Fig. 3. Architecture for network control delegation using ADVisor. Note that switches C, D and G are masked using virtual links

of network abstractions to end users. Note that the following architectures can be thought of as particular instantiations of our conceptual framework in Figure 1. The difference between the two architectures that follow is how the 'logic for exposing views to the users' creates views for the users.

A. *FlowVisor Based Architecture*

As mentioned earlier, we leverage existing mechanisms to achieve our goal of delegating some network control to end users. As our first attempt to provide users control over their flows we use FlowVisor. Figure 2 shows the proposed architecture using FlowVisor. As mentioned in II-C, an admin needs to first define policies in terms of flowspaces in the FlowVisor. These policies serve to tell FlowVisor, which controller can see which switches and which packets seen on those switches should be controlled by a particular controller.

As an example consider what happens when a host A_1 which is a VM belonging to User$_A$ generates a packet. The packet arrives at switch$_A$ and since switch$_A$ does not have any flow table entry, it does not know how to forward the packet. The packet is thus sent to FlowVisor. At this point FlowVisor checks to see the packet header and decides which flow-space or controller the packet belongs to. Since this packet should be under the control of User$_A$, FlowVisor forwards the packet to Controller$_A$ for making flow decision for this packet. Controller$_A$ on receiving this packet pushes new flow rules into switch$_A$. Note that before this flow rule is forwarded to the switches, it is inspected and vetted by the FlowVisor layer so as to ensure that Controller$_A$ does not inadvertently affect the packets under the flow-space of Controller$_B$.

The direct visibility of all switches in the path of users' flows permits to satisfy several local properties, as the architecture of the underlying network infrastructure is exposed to users with little mediation. The connectivity property is easily satisfied as long as there is connectivity in the underlying physical infrastructure. The datacenter consistency property is satisfied, as the exposed physical infrastructure matches the datacenter architecture. Multi-path is potentially possible, as long as the underlying physical network provides such a property.

In term of global properties, the non-interference among users is satisfied by design by mediating network flow. However, such an architecture fails to provide views satisfying the non-inference property, that is, preventing end users from mapping the complete underlying topology. This is true because if users collude with each other they could put their views of the topology and potentially discover the complete topology. For example, consider UserA and UserB in our example, though most of the switches that they control are not common to each other, sharing their views of the network would inevitably expose the whole topology.

B. *ADVisor Based Architecture:*

To overcome this limitation, in our second instantiation we replace FlowVisor with an instance of ADVisor as shown in Figure 3. The advantage of using ADVisor over FlowVisor is that we can abstract away some of the nodes in the physical topology and present a subset of the resulting view to the users. The intuition here is that even if a few cloud users collude with each other and share their views of the network, they will not be able to discover the exact underlying topology. Hence, ADVisor permits creating subsets of user views that satisfy the global property of non-inference. Note that if all the cloud users collude then nothing can be done to protect the network topology information but we argue that requiring more users to collude to discover the topology can mitigate the risk in practice. However, a limitation of this architecture is that it provides relatively less flexibility and control to end users as compared to the FlowVisor based architecture because now some of the switches have been abstracted away and become

part of virtual links and hence are not controllable by the users anymore. As the number of abstracted switches increases, the user views lose properties such as datacenter consistency or geographic consistency. Entire portions of the infrastructure might be seen as being part of the same network, while actual hosts might be distributed across data centers or geographic regions.

ADVisor is useful in providing virtual links to mask some of the physical switches on a given path between two physical switches but it will be more useful to have a mechanism to not only mask the physical switches as virtual links but to also add virtual switches in the Views. This will enable cloud providers to enrich the network view of the users while making sure that underlying topology isn't exposed completely. We believe that ADVisor can be extended to provide such a functionality. We leave this extension of ADVisor to future work.

C. Proof of Concept

To demonstrate feasibility and practicality of the framework instantiation, we constructed a prototype implementation for the FlowVisor-based architecture to enable separation of network controls among cloud users. This section briefly describes the implementation details.

The flowspace of each controller was specified using FlowVisor's policy language. Also, a topology that matches the topology in Figure 2 was emulated in mininet [8]. All the switches were directed to the port where FlowVisor was listening. Then a network slice was created for each user (UserA and UserB) through FlowVisor. Then, we configured the controllers for each user in a way that packets only travel over the switches specified in Figure 2 for each user. This was achieved by using a dictionary with keys being 3-tupled values composed of data path IDs (DPID), the physical port where packet arrived (in_port), and the source MAC address of the packet (packet.src). Based on this key, the dictionary gives the value of the output port to which the packet needs to be sent out. This application was implemented as a python module for POX controller. We were able to verify connectivity among the hosts in one slice and isolation among hosts of different slices using ICMP 'ping' messages.

D. Discussion

Instantiation examples above clearly demonstrate the need to balance two requirements – the degree to which users get control over their network flows, and the security concerns of the cloud provider. These two ends directly map to the two ends of the virtualization spectrum: at one end there is little or no virtualization, and switch resources in the network are directly sliced; and at the other end is the completely virtualized case where the topology visible to the cloud users is fully decoupled from the underlying infrastructure (this is the approach taken by Amazon EC2). The tradeoffs are different in these two approaches. While the virtualized view of the network provides users with an easy to use interface, it also makes it harder for them to efficiently and precisely schedule their flows. Furthermore, this approach is widely adopted by the current cloud providers because they don't have to give out any details regarding underlying topology, ensuring that they maintain any competitive edge as well as preventing users to infer any information about the network. On the other hand providing users transparency over the underlying network means that users will now have to take care of the failures in the network themselves and reroute the traffic when needed. The downside of this approach is that it may expose too much information about the cloud infrastructure and provide users control over physical resources raising security concerns. There is clearly a need to achieve a balance between these two approaches – adopting completely virtualized network deprives the users of the flexibility and control and is considered more secure from provider's viewpoint whereas fully non-virtualized view though gives more control to the cloud users, it may overwhelm users with the configuration tasks and may not be very secure. We believe that it will be beneficial to come up with metrics that can be used to quantitatively evaluate network control delegation approaches. We hope to work on developing these metrics and do a more detailed comparative evaluation of the two proposed architectures in future.

V. Related Work

We have already covered the role of ADVisor and FlowVisor in Section II. Both of them act as hypervisors for the network and allow multiple controllers to control the same network. ADVisor improves over the FlowVisor by providing virtualization of nodes along a path that connects two machines of a cloud user. Quantum [9] is an API developed for the OpenStack project. With this API, cloud provider can use different plugins to create a network for each tenant. Quantum is different from our work because it only provides cloud provider different methods to implement the network and hasn't been used yet to provide cloud users control over the physical network.

FlowN [10] is a recent work that proposes a fully virtualized solution for each tenant. This is in contrast to our goals where we are trying to achieve a vantage point by providing cloud users control over some switches and yet preventing the cloud infrastructure from cloud cartography attacks. Finally, RouteFlow [11] is another project that aims to provide routing algorithms to SDN users. RouteFlow can be thought of as a tool that can enhance SDN but it can not be used as standalone tool to provide different level of abstractions to cloud users and hence their work is orthogonal to ours.

Recently, Self Service Cloud (SSC) [12] computing was proposed to increase the flexibility for users to deploy useful services without having to rely on the cloud provider while at the same time reducing the access that the cloud provider has to client VMs. While this work focuses on the flexibility and user control over their VMs we focus on user control of their network flows.

Another recent work that is relevant is CloudWatcher [13], which uses OpenFlow to provide security monitoring services for cloud networks. CloudWatcher provides monitoring services for cloud infrastructure administrators to ensure that

their network packets are always inspected by pre-installed security devices (e.g., firewalls or intrusion detection systems). In contrast, our framework uses SDN to delegate configuration and monitoring capabilities to *end users*, providing some level of network abstractions without compromising the security of the infrastructure.

VI. Conclusion

In this paper we looked at the feasibility of providing cloud users more control over their network using abstractions provided by OpenFlow and software defined networks while balancing the concerns of cloud providers over giving users direct access to underlying physical infrastructure. We proposed a SDN based framework to enable delegation of some network control to end users and discussed various properties of the network views provided to users. We explored two architectures that can be used to expose the network to the cloud users and provided a brief overview of their pros and cons. We also presented a prototype implementation of one architecture.

For future work, we plan to further explore network control delegation to users and characterize more exhaustively different properties of network views that can be presented to end users and their security implications for the cloud provider. We also intend to study the interference effect of network control delegation and look at techniques to optimize bandwidth sharing among users in multi-tenant clouds.

Acknowledgment

This work has been partially supported by the Air Force Research Laboratory and the Air Force Office of Scientific Research under agreement number FA8750-11-2-0084.

References

[1] N. Leavitt, "Is cloud computing really ready for prime time?" *Computer*, vol. 42, no. 1, pp. 15 –20, jan. 2009.

[2] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, 2010, pp. 19–19.

[3] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, "Fresco: Modular composable security services for software-defined networks," in *To appear in the ISOC Network and Distributed System Security Symposium*, 2013.

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

[5] 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.

[6] E. Salvadori, R. Corin, A. Broglio, and M. Gerola, "Generalizing virtual network topologies in openflow-based networks," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. IEEE, 2011, pp. 1–6.

[7] U. Hlzle, "Openflow @ Google," Open Networking Summit, April 2012.

[8] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM, 2012, pp. 253–264.

[9] "Openstack quantum," http://wiki.openstack.org/Quantum.

[10] "Scalable network virtualization in software defined networks," http://www.cs.princeton.edu/~jrex/papers/ieeeinternet12.pdf.

[11] "Routeflow," https://sites.google.com/site/routeflow.

[12] S. Butt, H. A. Lagar-Cavilla, A. Srivastava, and V. Ganapathy, "Self-service cloud computing," in *Proceedings of the 2012 ACM conference on Computer and communications security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 253–264. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382226

[13] S. Shin and G. Gu, "CloudWatcher: Network Security Monitoring Using Openflow in Dynamic Cloud Network," in *7th Workshop on Secure Network Protocols (ICNP-NPSec)*, Austin, Texas, USA, October 2013.