Enhancing Network Security through Software Defined Networking (SDN)

Conference Paper · August 2016

DOI: 10.1109/ICCCN.2016.7568520

Lei Xu
Texas A&M University

Enhancing Network Security through Software Defined Networking (SDN)

Seungwon Shin
School of Computing, KAIST
Email: claude@kaist.ac.kr

Lei Xu, Sungmin Hong, Guofei Gu
SUCCESS Lab, Texas A&M University
Email: {xray2012, ghitsh, guofei}@cse.tamu.edu

***Abstract*—Software Defined Networking (SDN) is an emerging technology that attracts significant attention from both industry and academia recently. By decoupling the control logic from the closed and proprietary implementations of traditional network devices, it enables researchers and practitioners to design new innovative network functions/protocols in a much more flexible, powerful, and easier way. We believe SDN provides new research opportunities to security, and it can greatly impact network security research in many different ways. However, till today, SDN has not been well recognized by the security community yet. In this systematic survey on SDN security, we investigate how the new features provided by SDN can help enhance network security and information security process. By systematically reasoning the opportunities introduced by SDN to network security, we hope to provide new insights for future research in this important area.**

***Index Terms*—Software-Defined Networking, Network Security**

I. Introduction

Software Defined Networking (SDN) has quickly emerged as a new promising technology for future networks. With the separation of control plane from data plane thus enabling the easy addition of new, creative, powerful network functions/protocols, SDN has attracted significant attention from both academia and industry. In academia, since the publication of OpenFlow [33], which is a key component to realize the SDN concept, many research ideas based on SDN/OpenFlow have been proposed (and still go on) [35] [48] [6] [20] [26]. In industry, SDN is widely considered as the new paradigm for future networks, and many companies are deploying or plan to deploy such technology in order to strengthen their network architectures [24], reduce operational cost [19], and enable new network applications/functions [60].

The reason why many researchers and practitioners have interests in SDN is mainly because by decoupling the control logic from the closed, proprietary implementations of traditional network switch infrastructure, SDN enables us to design and distribute innovative flow handling and network control algorithms easily, and it helps us add much more intelligence and flexibility to the control plane. With the help of SDN, we can dynamically control network flows and monitor network status easily. For example, by employing SDN, we can easily implement a network load balancing function that is not easily and cheaply solved with existing techniques. These powerful and rich functions from SDN enable people to create new and creative network services or architectures. Some researchers propose a network virtualization service for a cloud network or a large-scale enterprise network [47] [37], and it has come into the spotlight due to its efficient resource management [17] [40]. In addition, a new wireless network architecture that can provide more robust wireless network services has been proposed based on SDN technology [38]. Besides these examples, there are many cases of employing SDN for a new service or a new architecture [35] [48] [6], and some have been already applied into real world network environments (e.g., Google data centers [24]).

Compared with the networking community, the security community is relatively slow in embracing SDN. As an evidence, while there are more and more SDN research papers appearing in top networking venues and several new SDN-focused conferences created recently (e.g., ACM SOSR [1]), there is still less attention from the security researchers. Why does this happen? Is it because SDN does not provide benefit for security applications/services? We argue that this is *not* true. As a matter of fact, we believe that SDN can, in time, prove to be one of the most impactful technologies to drive a variety of innovations in network security. To this end, we conduct a systematic study on the relation between SDN and security. In general, there are two high-level areas in SDN security research, i.e., (i) enhancing security using SDN, and (ii) studying the security issues (e.g., vulnerabilities) inside SDN itself. This paper focuses on the first area. In particular, we are interested in answering the following question: Can we (and how to) leverage the new features provided by SDN to enhance network security? The desire to answer this question forms the main motivations of this paper.

Overall, the main goals and contributions of this paper are two-fold:

- First, we systematically introduce the SDN technology to a broader range of security researchers. We believe the reason why the security community is slow in embracing SDN is mainly because it is currently not sufficiently exposed to them yet. A deeper understanding of the SDN technology will help security researchers produce new, interesting, and better security services or intelligent network defense systems.
- Second, we provide an in-depth investigation on how SDN features can bring benefits to security, illustrated with state-of-the-art research in the related areas. By demonstrating these new opportunities brought by SDN,

we hope to stimulate new creative ideas and more future work in this area.

II. Problem Statement

We believe that SDN can bring significant benefits to security research and it can also be combined with existing security research. We note that studying/enhancing the security threats/vulnerabilities/issues of SDN itself (e.g., [49, 41, 28, 31, 61, 21, 59, 44]) is *not* the focus of this paper. Instead, we want to systematically investigate the opportunities and challenges on how SDN can benefit network security. To this end, we start with reviewing the new features provided by SDN, then study how these features can enhance specific security functions. Finally, we will further discuss challenges in actually implementing security applications with SDN in the real world.

A. New Features Provided by SDN

SDN/OpenFlow provides programmability, dynamicity, flexibility, and intelligence to current network architectures, and its benefits can be delivered from four main features: (i) dynamic flow control, (ii) network-wide visibility with centralized control, (iii) network programmability, and (iv) simplified data plane.

Dynamic Flow Control: Based on SDN's basic characteristics (i.e., ask the control plane if the data plane does not have a flow rule to handle a network flow), a network application can control network flows dynamically. This feature is highlighted with network applications for flow control, such as dynamic load balancing [60] and network management application [2].

Network-Wide Visibility with Centralized Control: In SDN, all data planes are connected to a centralized control plane to receive control messages (e.g., flow rule insertion and data plane configuration). In addition, the control plane collects network status information from each data plane by sending a statistics query message. Therefore, a network application running on the control plane naturally has a view of all connected data plane, and it can control all data plane in a centralized way. Several network-wide monitoring applications with SDN (e.g., BigTap [5] and a network management application [27]) are good examples that benefit from this feature.

Network Programmability: Since all data planes in an SDN network can be controlled by a network application program, SDN provides a strong capability to *program* enable new network functions. This is similar to programming a smartphone (e.g., Android) app to enable unlimited creativity of functionalities. To empower this feature, several network programming languages have been proposed so far [56] [15], [4], [57], [36], and they help us program network functions easily.

Simplified Data Plane: Basically, the SDN architecture separates the data plane from the control plane, and thus the data plane only has relatively simple logic. This simplified data plane gives us chances of adding some new features NetFPGA [34], DevoFlow [9] are good examples of the simplified data plane and its modification.

B. Paper Overview

We now present the overview of this paper. We first describe the benefits that network security can obtain from each SDN feature, and we demonstrate with example security applications from state-of-the-art research. This systemization of knowledge is summarized in Table I. The first column denotes the features of SDN, and the second column presents their functions. The third and fourth columns show the benefits that each feature can provide to network security and example network *security* applications. The last column reviews the possible role of these SDN features in the classic defense-in-depth framework, i.e., prevention, detection, and response. Detailed explanation for each column will be presented in the following sections.

III. How Network Security Benefits from SDN Features?

In this section, we will investigate how the SDN new features, which have been discussed in the previous section, can provide benefits to network security. In addition, we will provide some example cases of security applications, which have been proposed previously or can be realized in the future, to help people understand their benefits.

A. Dynamic Flow Control

Benefit to Network Security: Controlling network flows dynamically provides many new possibilities in network security functions. First, we can implement a dynamic access control function, which is commonly used to protect a network. Previously, we need to install an independent middlebox (e.g., firewall) to achieve in-line access control. However, with the help of SDN, we do not need to set up additional middleboxes, but just use a network device (e.g., an OpenFlow switch/router) that supports SDN functions for access control. In addition, we can control network flows with diverse granularity (from 1 tuple to 12 tuples), and it enables us to control network flows more efficiently.

Second, it enables us to separate malicious (or suspicious) network flows from benign ones dynamically. This ability is quite useful when we want to differentiate security services. Suppose we simply monitor network flows to detect malicious (or suspicious) flows with a network intrusion detection system (NIDS). At this time, if an NIDS detects some flows and we want to investigate more about the flows, we may use in-depth security services (e.g., honeypot) to do it. In this case, we usually apply a proxy server to reroute or capture network flows for deeper investigation. However, if we apply SDN, we can simply build this function by controlling network flows dynamically. This case will be shown in the example case below (i.e., the case of intelligent honeypot).

Network Security Application Example: *[Firewall Example]* Implementing a firewall function in SDN is pretty simple

SDN Feature	Feature Description	Benefit to Security	Network/Security Application Examples	Roles in Defense
Dynamic flow control	SDN can control (e.g., reroute, forward, drop) network flows dynamically	Dynamically control malicious or suspicious network flows (packets), separate malicious network flows from benign flows	FlowVisor[47], OpenVirtex[3], FlowN[11], splendid[18], NVP[30], Random route mutation [12], Random host mutation[23], Varmour[55], FlowNAC[32], PBS[20]	Prevention, Response
Network-wide visibility with centralized control	All network status and flow information can be monitored and managed by a centralized server, which we call a controller	Monitor whole network in a centralized way for security services, detect network flooding or network anomaly efficiently and effectively (network-wide monitoring)	CloudWatcher[48], NetSecVisor [51], SIMPLE[43], FlowTags[14], OpenNF[16], SPHINX[10], DDoS detection/defense[6, 13, 25], Resonance[35], NetFuse[63], FleXam[52]	Detection, Response
Network programmability	SDN enables us to program network functions	Develop network security applications easily, open the gate of devising advanced network security applications	FRESCO[50], Nettle[56] , Frenetic[15], OpenSAFE[4], Procera[57], Controller Programming[36]	Detection, Response
Simplified data plane	SDN makes the data plane quite simple by moving out complicated control plane logic	Change the data plane lightweightly as a kind of security device by adding new modules	Avant-Guard[46], OFX[53], OpenSDWN[45]	Prevention, Detection, Response

TABLE I

OVERALL SUMMARY OF SDN FEATURES AND THEIR POTENTIAL CONTRIBUTIONS TO NETWORK SECURITY

and straightforward[1], and there are real firewall cases with SDN [55]. Figure 1 shows the implementation scenario of a firewall function with SDN/OpenFlow. When a switch receives a network packet (1), it reports it to an SDN controller if there are no flow rules to handle the packet, and the controller forwards this information to a firewall application (2). The firewall application first parses the received packet (3), checks whether the incoming packet violates security policies or not (4), and enforces a flow rule based on the policies (5). Finally, this rule is delivered to the switch by the controller (6). The switch puts the delivered flow rule into its flow table — in this example, the rule blocks the packet (7).

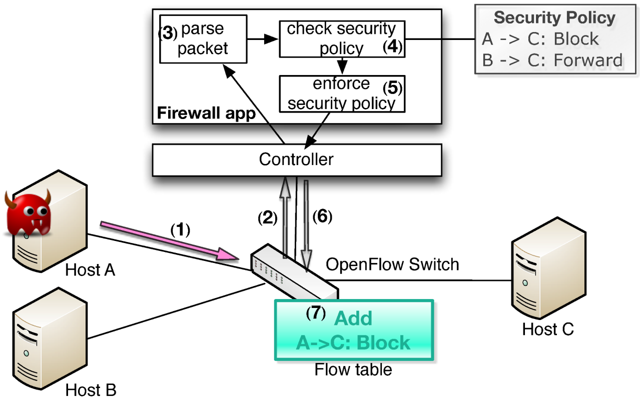

Fig. 1. Example firewall implementation with SDN.

This example scenario clearly shows that the dynamic flow control feature facilitates us to enforce firewall functionalities. Moreover, security policies can be easily modified by simply changing entries in a security table in Figure 1. Moreover, Varmour has announced its firewall product for an enterprise network by using the dynamic flow control feature[55].

[Intelligent Honeypot Example] When we find some suspicious (or malicious) network flows, we can handle them differently with the help of the dynamic flow control feature. It can be explained by showing an example scenario of implementing an intelligent honeypot architecture, shown in Figure 2. First, the attacker A sends a scan packet to the network port 443 of the target host B (S-1), and then such information is delivered to the controller and the intelligent honeypot application (A). Inside the application, four modules are working (P-1 to P-4), and the P-2 module tries to investigate whether a flow is suspicious or not. Since this is a normal TCP request (NO from P-2), it simply enforces a flow rule to forward a packet through the P-3 module and the rule is inserted into the flow table (F-1). However, since host B does not open port 443, host B simply returns a RST packet (R). Then, this information is delivered to the application. Now, the application infers that host A is likely scanning host B[2]. Thus, if the attacker A sends another scan packets to the port 445 (S-2), the application redirects this packet to a honeypot H dynamically (H).

This scenario also shows the benefit of the dynamic control feature. To implement this kind of application without SDN, we need to install a proxy server that can change network packets dynamically, and a complicated application is required to operate the proxy server as well. However, using SDN, we can implement this function by creating a relatively simple network application.

[Network-level Access Control Examples] SDN can also control accesses to network entities at the right level of privileges and policies according to network user and network applications. In the literature, FlowNAC proposes a fine-grained flow-based network access control in service level [32]. Furthermore, Hong et al. proposes Programmable

[1] In this paper, we simply focus on the basic function of a firewall (i.e., drop unwanted connections), because it is supported by all firewall products.

[2] Note this is only a simplified synthetic example. In practice, more evidences (e.g., more failed scan attempts) may be needed to infer a malicious entity.

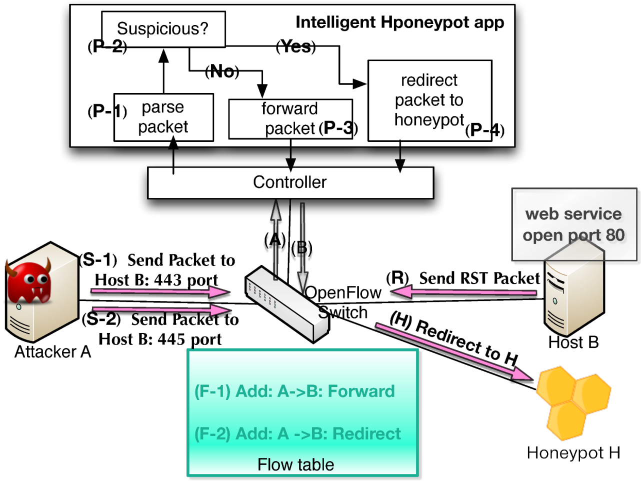

Fig. 2. Example intelligent honeypot implementation with SDN

BYOD Secrutiy [20], which embraces SDN techniques to provide a fine-grained access control upon application-specific network flows in BYOD (Bring Your Own Device) scenarios. Moreover, with the dynamic flow control capability, we still consider SDN has a potential to secure the entity accesses in other network scenarios, such as Internet of Things (LoT) and Wireless Sensor Networks (WSN).

[Network Separation Examples] Another network security application example, which can show the benefit of the feature of dynamic flow control, is a network separation application. In traditional networks, the very basic and simple way to separate a network is employing VLAN (Virtual LAN) technique [54], which adds specific IDs in a packet header (12-bits VLAN ID field) to differentiate packets for each tenant/user. However, VLAN technology incurs scalability issues in large-scale networks, such as data center, since it can only assign 4,096 different virtual networks. Also, typical static network separation in practice begets error-prone, manual burden of reconfiguration upon dynamic network/policy changes, which falls short of promptly reactive action for security purpose.

SDN communities catch this problem and propose several solutions [47, 18, 3, 11, 30]. Namely, SDN-based separation solutions provide the capability of different level abstractions with desired security properties, which not only separates the network segments efficiently at scale but veils the physical view of networks to users. One representative network separation example is FlowVisor [47], which is not dependent on some specific network identification fields to separate networks, instead it virtually isolates networks using OpenFlow functions. Therefore, theoretically, there is no upper bound in creating virtual networks with FlowVisor. To create a virtual network for each tenant, FlowVisor receives a network configuration policy from each tenant, and it creates routing paths based on each tenant's configuration. At this time, network flows for tenant A are not forwarded to networks for other tenants. Thus, the system guarantees that each tenant's network is virtually separated from others.

[Moving Target Defense Examples] The flexibility of SDN also empowers network participants to conduct a moving target defenses against network attacks with dynamic control features. Jafarian et al. have proposed a new network architecture that randomly mutates IP addresses of hosts, and thus it makes an attacker hard to find a target host [23]. Duan et al. proposes Random Route Mutation [12] to enable dynamic change of route to defend against network attacks including DoS, eavesdrop, and reconnaissance.

B. Network-Wide Visibility with Centralized Flow Control

Benefit to Network Security: Network-wide monitoring is an important and necessary function in network security. Traditionally, to monitor entire subnets (including both through traffic and internal traffic), we need to install or set up monitoring sensors and collect network information in *every* network device or link, which turns out to be not easy to realize in large-scale networks in the real world.

Using SDN can ease the network-wide monitoring and the detection/defense of network-wide attacks. Based on the SDN's basic characteristics (i.e., control and monitor whole networks in a centralized way), we can monitor each network device easily by collecting network statistics information from them and receiving flow request messages from each network device. Network statistics information can be easily captured by sending request messages from a network application, and a network application can understand overall network topology and routing information by analyzing flow requests from network devices. The holistic network view also facilitates the detection and defense of network attacks. The network administrator can adopt anomaly analysis to pinpoint network-wide attacks by monitoring the network state change. Moreover, he/she can reorganize and tune network resource to mitigate those large-scale network attacks.

This feature can also improve the utilization of security devices by assigning specific network flows to necessary/specific security appliances, e.g., hardware devices, middleboxes, and virtual network functions. In a complicated network environment, it is not easy to configure a network architecture to let all (or most) network flows be monitored by certain security appliances, because some (or many) network flows are hard to be delivered to the installed (physically fixed) location of security appliance. In this case, with the help of the network-wide visibility of SDN, we can understand where network flows are passing and where security appliances are installed, and we can reroute network flows to make them pass through certain required network security appliances. We show later how this feature can improve the utilization of security appliances by presenting the approach of the recent research work [48, 43, 14].

Network Security Application Example: *[Network-Wide Flow Monitoring Examples]* We can write a simple network application that monitors multiple network devices and detect attacks, and this example application is shown in Figure 3. This application consists of 5 modules (P-1 to P-5), and each module conducts the following operations. P-1 module sends a request for network status information to each network device frequently, P-2 module receives a response of network status from each device, P-3 module analyzes the collected status information, P-4 module detects some anomalous network flows, which will be considered as attack flows, and P-5 module finally enforces a flow rule to block detected flows. Here, we can observe that this application can easily collect network status information by simply sending a request message (i.e., an OpenFlow message).

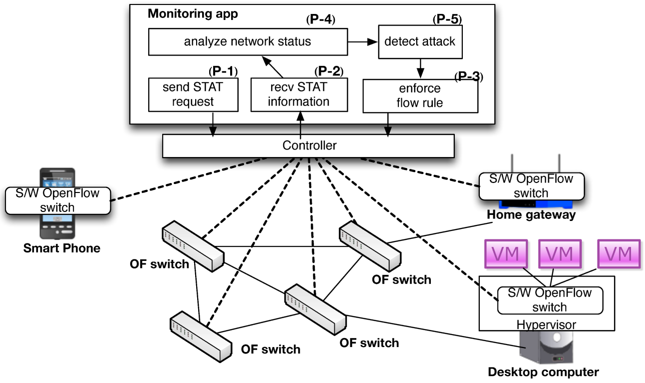

Fig. 3. Example Network-Wide Flow Monitoring Application with SDN.

In this scenario, the more interesting thing is that we can not only monitor network switches, but also other devices, such as a home gateway system, a hypervisor installed in a desktop computer, and a smart phone. Since the data plane for SDN functions (e.g., OpenFlow switch) can also be implemented as a software program, we can install this program into most devices for network communication, such as home gateway [7], hypervisor [58], and smart phone [62]. Therefore, a network-wide monitoring function can be realized in including most network related devices, and we believe that it can improve the effectiveness of and efficiency of security monitoring systems. Moreover, for efficient information collection, FleXam [52] proposes a sampling extension to OpenFlow protocol to facilitate security applications.

[Network Attacks Detection/Defense Examples] With holistic network view and centrality of network control logic, SDN provides powerful solutions to monitor and detect network attacks by collecting useful network information (i.e., statistics, control messages) and allowing security applications to take instant, smart actions on their own logic. SPHINX [10] presents a flow-graph model learned from SDN/OpenFlow messages to detect various network-level attacks on network topology and the data plane forwarding. Braga et al. have suggested an application that monitors network flows to detect a network flooding attack with OpenFlow [6]. NetFuse [63] monitors network devices to find some suspicious network flows with this feature.

SDN also exhibits a potential to tackle infamous network-level attacks, such as DDoS attacks, since SDN can catch important information from the entire network in a timely manner and facilitate defense algorithms customized/programmable to detect such attacks. Bohatei [13] proposes to mitigate several DDoS attacks by leveraging the flexibility of SDN. SPIFFY [25] utilizes holistic topology view of SDN to provide a Temporary Bandwidth Expansion (TBE) scheme to detect bots blamed for Link Flooding attacks.

[Security Appliances Deployment Examples] SDN can also leverage the global network view for more advanced security enforcement by providing more network information and primitives to better deploy/place/control security appliances and virtual functions, such as Firewall, Deep Packet Inspection (DPI). CloudWatcher [48] and NetSecVisor [51] provide approaches to force network flows under the inspection of certain required network security appliances. SIMPLE [43] and FlowTags [14] enforce security policies by efficiently steering traffic to security middleboxes. Currently, SDN is an important complementary to NFV to enforce security policies. [13] also showcases this point by utilizing SDN to steer suspicious traffic to defense VMs running security virtual functions. OpenNF [16] provides a new control plane architecture that supports coordinated controls of the internal state of security virtual functions and network forwarding state through its own primitives/APIs, thereby maintaining the up-to-date security virtual functions dynamically along with flow control without losing performance.

C. Network Programmability

Benefit to Network Security: Network security functions are usually built by deploying some hardware middleboxes or installing some software programs. They have predefined functions for network security, and commonly it is not easy to change or modify these functions. However, sometimes, it is hard to predict necessary security functions for a network. Then, in this case, what if it turns out that deployed security boxes are not so necessary to secure a network but need to install a different security function? We could dispose of the old boxes and buy a new box, which causes additional cost.

The network programmability feature of SDN can help us in this situation by enabling us to program network security functions easily. For example, we can create a network scan detection SDN application, and we can even implement an intelligent network security application in SDN control plane, such as a DDoS detection application. Programming network security applications is very useful and cost effective, because we do not need to buy additional hardware boxes or software programs to deploy network security services, but create and deploy network security applications running on a controller.

Network Security Application Example: *[Network Application Programming Examples]* Shin et al. have proposed a new framework (named FRESCO [50]) for creating security applications with SDN. This framework provides a script language and a modular composable programming model to help programmers easily develop SDN network security applications. Suppose a network administrator wants to implement a network intrusion detection system (NIDS) with a FRESCO script. He simply writes a script to coordinate several modules to compose a network intrusion detection function. Likewise, we can write scripts to make any desired network security applications, such as network scan detection and reflector networks [50]. We also note several works [15, 56, 4, 57, 36] propose to enhance the programming paradigm of SDN application, which facilitate network administrators to enforce their security policies in an efficient, secure and cost-effective manner.

D. Simplified Data Plane

Benefit to Network Security: Compared with the previous generation network devices, the hardware (i.e., the data plane) for SDN can be easily modified, because it consists of relatively simple hardware modules and moves out complicated control plane modules. It provides a chance of extensions of new network functions (e.g., [9]). This concept can also be applied to security. We can extend the data plane of SDN to make it more suitable for security purposes.

Network Security Application Example: *[Flexible Data Plane Security Extensions]* To date, researchers have proposed several new data plane architectures for SDN community to make the data plane suitable for security usage. Avant-Guard [46] adds some new simple components (e.g., logic for migrating TCP connections), which make the data plane more scalable and provide new security functionalities. OFX [53] extends SDN/OpenFlow switches with customized security functionalities and enables the control plane to manage those addon security features. OpenSDWN [45] further extends a wireless access point as SDN/OpenFlow switch to better control the wireless transmission with virtual middleboxes inside the data plane of Click software router [29]. Despite the non-trivial extension to the data plane, it opens up the new opportunities of SDN security applications for wireless networks.

E. Final remark

While we discuss each feature individually, we acknowledge that in reality, it is usually hard to separate them for specific security applications. For example, distributed firewall [22] can naturally combine multiple features, e.g., dynamic flow control and network-wide visibility with centralized control. We envision future SDN security applications could combine any set of SDN features to achieve their desired capabilities.

IV. How SDN features Enhance Information Security Process?

Previously, we have discussed how SDN features benefit network security and we will now address how these features enhance the information security's basic triad: (i) prevention, (ii) detection, and (iii) response, which is a basic, well-known framework to enable defense in depth.

A. Prevention

Prevention is a process to stop attackers from contacting targets for protection, and usually it is realized by setting some security policies that define who (or what) can (or cannot) access whom (or what). This process requires careful planning and investigation to minimize mistakes because it is possible that security policies block benign users or accept malicious ones. Therefore, determining security policies is the most important job, and security policies are usually not changed after they are decided. However, many existing network architectures are quite complicated and varying continuously, and thus access control based on static policies may not be enough to protect and manage a large, dynamic network [35]. In this case, we need a dynamics access control method, which some recent network devices support [8], and it typically requires the installation of additional middleboxes into a network.

The dynamic flow control feature of SDN can enhance the prevention process by realizing dynamic access control functions without adding middleboxes. With the help of this feature, we can virtually turn each network device into a network security device that can prevent network attacks dynamically. As shown in the previous firewall example in Figure 1, we can simply change security policies dynamically by modifying a security policy table, and the changed policies are automatically enforced when the data plane asks for a rule to handle a network flow. It makes our network management simple and efficient.

However, although this feature enhances the prevention process, we could face another new problem, which is called *dynamic flow tunneling* [41]. Unless we use this feature carefully, it is possible that the dynamic flow control feature could let a malicious flow evade defined access control policies. This evasion scenario is shown in Figure 4. We assume that there is a buggy load balancing application that changes packet headers, and there is a firewall that blocks all connection attempts from host A to host B. In this case, when host A sends a packet to host B (1), this information is delivered to the application (2) and the application processes a particular logic and enforces flow rules (3 - 6). When the enforced flow rules change the source IP address and the destination IP address (7), the final packet is changed as a packet sent from host D to host B. Since this packet does not violate the security policy of the firewall, the packet is finally delivered to host B.

This issue has been revealed by Porras et al. [41], and some controllers (e.g., Floodlight and NOX) add a patch to address this issue [39] [41]. However, we have not heard news that other network controllers (e.g., POX [42]) address this issue. If one wants to use the dynamic flow control feature to build a dynamic access control method, one needs to keep in mind this issue not to make additional security holes.

B. Detection

Detection is a basic security process to discover network intrusions, and two types of detection methods are commonly used: (i) misuse detection and (ii) anomaly detection. Misuse detection detects attacks based on known patterns (a.k.a., signature), and anomaly detection finds attacks by finding malicious patterns in network traffic that do not conform to expected normal behaviors.

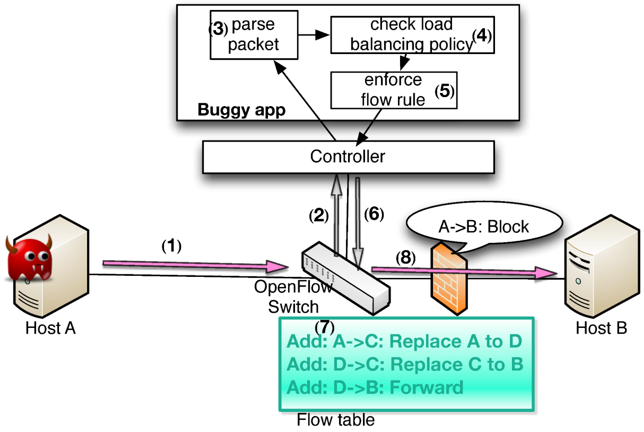

Fig. 4. Example Dynamic Flow Tunneling Scenario.

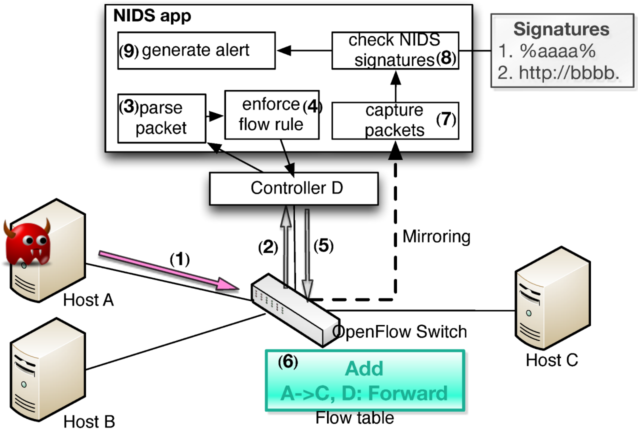

Fig. 5. Example SDN of Implementation of Network Intrusion Detection System.

In the case of anomaly detection, the network-wide visibility with centralized control feature can enhance the performance of detection because this feature enables a detection system to monitor all (or most of) network devices and thus can have a global view of network status, which provides much more information than local views. For example, to detect flooding attacks, a prompt detection system needs to monitor as many network links as possible because a flooding attack usually comes from many different sources (e.g., botnets). In this case, thanks to the feature of network-wide visibility with centralized control, we can easily achieve the goal in SDN.

Moreover, SDN/OpenFlow can also contribute to misuse detection with the network information support (specifically, packet payload). A intuitive way in current SDN/OpenFlow is to instruct the data plane to pass all network packets with their payloads to the SDN control plane, in which the SDN control plane can conduct some in-depth inspections. However, we consider such solution is not practical/efficient in scalability. In this case, we consider a smart way to realize the misuse detection, as an NIDS illustrated in Figure 5. When an attacker sends a packet (1), the data plane delivers this information to the control plane since it is a new flow (2). Then, the NIDS application parses the information (3), and enforces a flow rule to realize network mirroring function if it considers the network flow is suspicious(4, 5). In this case , the flow rule forwards the following packets to two network ports; (i) toward original target host and (ii) toward the control plane for mirroring (6). Then, all packets (including payloads) are delivered to the control plane and the NIDS application (7). This application checks delivered packets by inspecting whether the packets include any patterns defined in signatures (8). If it can find any packets that match some signature, it generates alerts (9).

In terms of the role of each SDN feature in the detection process, the network-wide visibility with centralized control can allow us to implement distributed network intrusion detection system easily. The dynamic flow control can be used for collecting packet information efficiently, and we can devise an advanced intrusion detection system with the network programmability feature.

C. Response

Response to attacks (e.g., attack mitigation) is an essential part in the defense-in-depth security framework. However, historically it is the toughest one to actually achieve. Traditionally, to respond to attacks, we may install middleboxes that drop or reject attack trials and/or isolate/quarantine compromised hosts to protect other hosts in the network. The dynamic flow control feature of SDN can be used to significantly enhance this process. Detected attack trials can be easily dropped by this feature, and network isolation or quarantine can be also easily implemented with SDN.

The network programmability feature can enrich the response functionality dramatically because various flow handling schemes for response (e.g., network isolation and quarantine) can be easily implemented with this feature. For example, with FRESCO [50], which we have introduced in the previous section, we can quarantine or isolate infected hosts (or networks) by simply composing modules with tens of lines of scripts. This makes the response process, traditionally recognized as a difficult job, simple and easy.

Readers may notice that the simplified data plane feature has not been mentioned yet. This is mainly because this feature does not directly provide new functions but instead provides a possibility of adding new functions, which can bring benefits to the entire prevention-detection-response framework. For example, in order to enhance prevention (of information leak), we may add encryption/decryption operation components into the SDN data plane. Similarly, for enhancing detection, we many add a component that can hold simple intrusion signatures into the data plane.

V. Discussion

The main goal of this paper is to draw some reasonable answers to our main research question - *can we (and how to)* *leverage the new features provided by SDN to enhance network security?*. Based on our serious surveys and in-depth analysis of SDN features and their applications discussed in this paper, we claim that SDN can clearly enhance network security functions in the following points. First, its ability of controlling network flows dynamically can provide more flexible deployments of security functions on a network because it allows us to enable security functions on SDN-enabled network devices without installing additional devices (e.g., middleboxes). Second, its network-wide visibility can realize network-wide monitoring in terms of security. This ability provides a holistic view to us, and thus we can comprehend network attacks widely distributed in the Internet (e.g., network-wide scanning or DDoS) much more efficiently than legacy network monitoring systems. Third, its programmability helps us develop more advanced network security functions. We can (relatively) easily implement a prototype security system without putting much effort. As such, SDN features can be leveraged in accelerating the development of new and advanced network security functions.

VI. Conclusion

In this paper, we introduce the SDN technology and systematically investigate its usage for security. Although many people have interests in this technology, until now, it is not yet well embraced by security researchers. We believe that SDN can, in time, prove to be one of the most impactful technologies to drive a variety of innovations in network security. We hope this study can not only provide a quick introduction and systematic survey but also give significant insights for using SDN for better security applications and stimulate more future research in this important area.

References

[1] ACM. Acm sigcomm symposium on sdn research (sosr). http://www.sigcomm.org/events/SOSR.

[2] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: Dynamic Flow Scheduling for Data Center Networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, 2010.

[3] Ali Al-Shabibi, Marc De Leenheer, Matteo Gerola, Ayaka Koshibe, Guru Parulkar, Elio Salvadori, and Bill Snow. OpenVirteX: Make Your Virtual SDNs Programmable. In *HotSDN'14*, 2014.

[4] Jeffrey R. Ballard, Ian Rae, and Aditya Akella. Extensible and Scalable Network Monitoring Using OpenSAFE. In *Usenix INW/WREN*, 2010.

[5] BigSwitch. Bigtap: Monitor traffic everywhere. http://www.bigswitch.com/products/big-tap-network-monitoring.

[6] R. S. Braga, E. Mota, and A. Passito. Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow. In *Proceedings of the 35th Annual IEEE Conference on Local Computer Networks*, LCN, 2010.

[7] Kenneth L. Calvert, Rebecca E. Grinter, W. Keith Edwards, Ye Deng, Nick Feamster, and Xuzi Zhou. Instrumenting Home Networks. 2010.

[8] Cisco. Cisco ios security: Access control lists. http://www.cisco.com/en/US/docs/ios/12_2/security/configuration/guide/scfacls.html.

[9] Andy Curtis, Jeff Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. DevoFlow: Scaling Flow Management for High-Performance Networks. In *Proceedings of ACM SIGCOMM*, 2011.

[10] Mohan Dhawan, Rishabh Poddar, Kshiteej Mahajan, and Vijay Mann. SPHINX: Detecting Security Attacks in Software-Defined Networks. In *NDSS'15*, 2015.

[11] D.A. Drutskoy. Software-defined network virtualization with flown. Master Thesis, 2012. ftp://ftp.cs.princeton.edu/techreports/2012/929.pdf.

[12] Qi Duan, Ehab Al-Shaer, and Haadi Jafarian. Efficient Random Route Mutation considering flow and network constraints. In *CNS'13*, 2013.

[13] Seyed K. Fayaz, Yoshiaki Tobioka, Vyas Sekar, and Michael Bailey. Bohatei: Flexible and Elastic DDoS Defense. In *Usenix Security'15*, 2015.

[14] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul. Enforcing network-wide policies in the presence of dynamic middlebox actions using FlowTags. In *NSDI'14*, 2014.

[15] Nate Foster, Michael J. Freedman, Rob Harrison, Jennifer Rexford, Matthew L. Meola, and David Walker. Frenetic: a high-level language for OpenFlow networks. In *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*, 2010.

[16] Aaron Gember-Jacobson, Raajay Viswanathan, Chaithan Prakash, Robert Grandl, Junaid Khalid, Sourav Das, and Aditya Akella. OpenNF: Enabling Innovation in Network Function Control. In *SIGCOMM'13*, 2014.

[17] Gigaom. Why Network Virtualization is Important. http://gigaom.com/2009/02/02/why-network-virtualization-is-important/.

[18] Stephen Gutz, Alec Story, Cole Schlesinger, and Nate Foster. Splendid Isolation: A Slice Abstraction for Software-Defined Network. In *HotSDN'12*, 2012.

[19] Brandon Heller, Srini Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. ElasticTree: Saving Energy in Data Center Networks. In *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation*, NSDI, 2010.

[20] Sungmin Hong, Robert Baykov, Lei Xu, Srinath Nadimpalli, and Guofei Gu. Towards SDN-Defined Programmable BYOD (Bring Your Own Device) Security. In *NDSS'16*, 2016.

[21] Sungmin Hong, Lei Xu, Haopei Wang, and Guofei Gu. Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures. In *NDSS'15*, 2015.

[22] Sotiris Ioannidis, Angelos D. Keromytis, Steve M. Bellovin, and Jonathan M. Smith. Implementing a distributed firewall. In *Proceedings of the 7th ACM conference on Computer and communications security*, 2000.

[23] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi 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*, HotSDN '12, 2012.

[24] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. B4: Experience with a Globally-deployed Software Defined Wan. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, 2013.

[25] Min Suk Kang, Virgil D. Gligor, and Vyas Sekar. SPIFFY: Inducing Cost-Detectability Tradeoffs for Persistent Link-Flooding Attacks. In *NDSS'16*, 2016.

[26] Ahmed Khurshid, Wenxuan Zhou, Matthew Caesar, and P. Brighten Godfrey. VeriFlow: verifying network-wide invariants in real time. In *NSDI'13*, 2013.

[27] Hyojoon Kim and N. Feamster. Improving network man-

agement with software defined networking. *Communications Magazine, IEEE*, 2013.

[28] Rowan Kloti, Vasileios Kotronis, and Paul Smith. OpenFlow: A Security Analysis. In *Proceedings of the 8th Workshop on Secure Network Protocols (NPSec'13)*, October 2013.

[29] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The Click Modular Router. *ACM Trans. Comput. Syst.*, 2000.

[30] Teemu Koponen, Keith Amidon, Peter Balland, Martn Casado, Anupam Chanda, Bryan Fulton, Igor Ganichev, Jesse Gross, Natasha Gude, Paul Ingram, Ethan Jackson, Andrew Lambeth, Romain Lenglet, Shih-Hao Li, Amar Padmanabhan, Justin Pettit, Ben Pfaff, , Rajiv Ramanathan, Scott Shenker, Alan Shieh, Jeremy Stribling, Pankaj Thakkar, Dan Wendlandt, Alexander Yip, and Ronghua Zhang. Network Virtualization in Multi-tenant Datacenters. In *NSDI'14*, 2014.

[31] Diego Kreutz, Fernando M. V. Ramos, and Paulo Verissimo. Towards Secure and Dependable Software-Defined Networks. In *Proceedings of ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'13)*, August 2013.

[32] Jon Matias, Jokin Garay, Alaitz Mendiola, Nerea Toledo, and Eduardo Jacob. FlowNAC: Flow-based Network Access Control. In *3rd European Workshop on Software-Defined Networks*, 2014.

[33] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38, March 2008.

[34] Jad Naous, Glen Gibb, Sara Bolouki, and Nick McKeown. NetFPGA: reusable router architecture for experimental research. In *Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, 2008.

[35] Ankur Nayak, Alex Reimers, Nick Feamster, and Russ Clark. Resonance: Dynamic Access Control for Enterprise Networks. In *Proceedings of WREN*, 2009.

[36] Tim Nelson, Arjun Guha, Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. A Balance of Power: Expressive, Analyzable Controller Programming. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, 2013.

[37] Nicira. Network virtualization platform. http://nicira.com/en/network-virtualization-platform.

[38] Openflow.org. Pantou. http://www.openflow.org/wk/index.php/Pantou_:_OpenFlow_1.0_for_OpenWRT.

[39] OpenFlowSec.org. Se-floodlight. http://www.openflowsec.org/Technologies.html.

[40] Oracle. The Growing Importance of Network Virtualization. https://blogs.oracle.com/drcloud/entry/the_growing_importance_of_network.

[41] Philip Porras, Seungwon Shin, Vinod Yegneswaran, Martin Fong, Mabry Tyson, and Guofei Gu. A Security Enforcement Kernel for OpenFlow Networks. In *Proceedings of the first workshop on Hot topics in software defined networks*, HotSDN '12, 2012.

[42] POX. Python network controller. http://www.noxrepo.org/pox/about-pox/.

[43] Zafar Ayyub Qazi, Chent-Chun Tu, Luis Chiang, Rui Miao, Vyas Sekar, and Minlan Yu. SIMPLE-fying Middlebox Policy Enforcement Using SDN. In *Sigcomm'13*, 2013.

[44] Christian Rpke and Thorsten Holz. SDN Rootkits: Subverting Network Operating Systems of Software-Defined Networks. In *RAID'15*, 2015.

[45] Julius Schulz-Zander, Carlos Mayer, Bogdan Ciobotaru, Stefan Schmid, and Anja Feldmann. OpenSDWN: Programmatic Control over Home and Enterprise WiFi. In *SOSR'15*, 2015.

[46] Seungwon Shin and Vinod Yegneswaran and Phil Porras and Guofei Gu. Avant-guard: Scalable and vigilant switch flow management in software-defined networks. In *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS13)*, November 2013.

[47] R Sherwood, G Gibb, K K Yap, and G Appenzeller. Can the production network be the testbed. In *Proceedings of USENIX Operating System Design and Implementation*, OSDI, 2010.

[48] Seungwon Shin and Guofei Gu. CloudWatcher: Network Security Monitoring Using OpenFlow in Dynamic Cloud Networks (or: How to Provide Security Monitoring as a Service in Clouds?). In *Proceedings of the 7th Workshop on Secure Network Protocols (NPSec12), co-located with IEEE ICNP12*, October 2012.

[49] Seungwon Shin and Guofei Gu. Attacking Software-Defined Networks: A First Feasibility Study (short paper). In *Proceedings of ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'13)*, August 2013.

[50] Seungwon Shin, Phil Porras, Vinod Yegneswaran, Martin Fong, Guofei Gu, and Mabry Tyson. FRESCO: Modular Composable Security Services for Software-Defined Networks. In *Proceedings of the 20th Annual Network and Distributed System Security Symposium (NDSS'13)*, February 2013.

[51] Seungwon Shin, Haopei Wang, and Guofei Gu. A First Step Toward Network Security Virtualization: From Concept To Prototype. In *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY 2015*, 2015.

[52] Sajad Shirali-Shahreze and Yashar Ganjali. FleXam: Flexible Sampling Extension for Monitoring and Security Applications in OpenFlow. In *HotSDN'13*, 2013.

[53] John Sonchack, Adam J. Aviv, Eric Keller, and Jonathan M. Smith. Enabling Practical Software-defined Networking Security Applications with OFX. In *NSDI'16*, 2016.

[54] IEEE Standard. 802.1q vlan. http://www.ieee802.org/1/pages/802.1Q.html.

[55] VArmour. http://www.varmour.com/.

[56] Andreas Voellmy and Paul Hudak. Nettle: Taking the Sting Out of Programming Network Routers. In *PADL*, 2011.

[57] Andreas Voellmy, Hyojoon Kim, and Nick Feamster. Procera: A Language for High-level Reactive Network Control. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, 2012.

[58] Open vSwitch. An open virtual switch. http://openvswitch.org/.

[59] Haopei Wang, Lei Xu, and Guofei Gu. FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks. In *DSN'15*, 2015.

[60] Richard Wang, Dana Butnariu, and Jennifer Rexford. OpenFlow-Based Server Load Balancing Gone Wild. In *Proceedings of Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, HotICE, 2011.

[61] Xitao Wen, Yan Chen, Chengchen Hu, Chao Shi, and Yi Wang. Towards A Secure Controller Platform for OpenFlow Applications (short paper). In *Proceedings of ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'13)*, August 2013.

[62] Kok-Kiong Yap, Te-Yuan Huang, Masayoshi Kobayashi, Yiannis Yiakoumis, Nick McKeown, Sachin Katti, and Guru Parulkar. Making use of all the networks around us: a case study in android. *SIGCOMM Comput. Commun. Rev.*

[63] Vishal Singh Cristian Lumezanu Geoff Jiang Ye Wang, Yueping Zhang. NetFuse: Short-circuiting Traffic Surges in the Cloud. *ICC, IEEE*, 2013.