Toward Secure Software Defined Networks, A Look at Attacks and Adversaries

Akshay Viswakumar
Student Number: 32971665
Electrical and Computer Engineering
University of British Columbia
Vancouver, British Columbia V6T 1Z4
Contact: https://www.linkedin.com/in/akshayviswa/

***Abstract*—Software Defined Networking (SDN) represents a paradigm shift in computer communication. It offers a means of simplifying network deployment and management. It also provides enhancements that will aid the creation of next generation cellular networks. More network operators and service providers are beginning to rely on SDN. The transition from conventional networking to SDN is inevitable and as a technology that will eventually govern communication over the internet, it is important that SDN is hardened against vulnerabilities either intrinsic or inherited from conventional networking. Through this study I have analyzed, categorized and developed upon prior research carried out in relation to the security of SDN. To my knowledge, this is also the first review in SDN to have focused on the evaluating real attacks proposed by researchers in recent times. Based on the evaluation, I created an adversary profile which helped provide insights about how an SDN could be defended effectively. I also discovered flaws with existing SDN specific security models which were unable to consider attacks that did not exploit visible attack surfaces[1].**

I. Introduction

There has always been a need for easy and efficient ways to deploy, monitor and operate computer networks. Conventional networking equipment didn't help the cause with its use of separate hardware that had unique software stacks for each network function (routing, switching, load balancing, firewalling etc).

Researchers have been proposing ideas for programmable networks since the mid 1990s. Feamster *et al.* have chronicled the intellectual evolution of programmable networks in their paper [1]. They divide the chronology into three distinct stages based on breakthroughs in research that occurred during each stage:

(1) The period between 1990 and 2000 saw proposals such as the SwitchWare architecture [2] that called for the creation of abstraction layers in network control devices based on function.

(2) In the years between 2001 and 2007 researchers called for controller based architectures such as SoftRouter [3] and Path Compute Element (PCE) [4] which expounded the benefits of logically separating the control plane and forwarding plane within the network.

[1]Attack surface refers to components of a system, vulnerable or otherwise, that an adversary may target to launch an attack.

(3) Finally, between 2007 and 2010 came the birth of OpenFlow [5] which was the first framework that could practically realize core ideas from past stages.

The term "Software Defined Networking" (SDN) was coined in 2009 when Greene used it in [6] to describe the broad capabilities of OpenFlow. Since then, SDN has seen usage in research related to programmable networks. The term has also become a buzzword in the networking industry where it's not uncommon these days to see it used imprecisely, especially in marketing materials. Therefore, in the context of this work, I would like to state that the term SDN will be used in-line with its definition as stated in RFC7426 [7]. That is, SDN is "*a programmable networks approach that supports the separation of control and forwarding planes via standardized interfaces*".

SDN has been steadily gaining prominence since the creation of OpenFlow. Today, large network operators have entire backbones of their data center networks (like Google B4 [8]) that run entirely using SDN. Further, the dynamism and flexibility offered by SDN have prompted researchers to propose its use in 5G cellular networks to facilitate use cases such as network slicing [9] which many believe will pave the way toward Enhanced Mobile Broadband (eMBB), Massive Machine Type Communications (mMTC) and Ultra Reliable and Low Latency Communication (uRLLC) as outlined in the International Telecommunication Union (ITU) IMT-2020 goals. [10]

SDN is poised to play a key role in the evolution of networking in the years ahead. As more operators and service providers switch to SDN based architectures, the amount of internet traffic carried over software defined networks will also increase. This poses the question about the security of a software defined network. Being radically different from conventional networks the software defined network may have vulnerabilities, either inherited or entirely new, that malicious actors may exploit to cause disruption. There is thus a need to keep scrutinizing SDN from the perspective of cybersecurity so that as many issues may be exposed and rectified before its widespread adoption.

For this study I performed an analysis of 20 different attacks that SDN researchers have proposed in recent times. While each attack was unique, there were plenty of commonalities that I could derive based on the adversary's motives and

capabilities. The characteristics of the adversary were used to create a detailed adversary profile based on key parameters like detectability, location and initial capabilities. This kind of an adversary profile, based on actual attacks, is the first of its kind and It helps provide good insights about practical defensive measures that need to be considered to safeguard a software defined network.

Security models help define generic threat vectors. SDN specific security models have been the norm for SDN security research since they could theoretically help predict any kind of attack. Unfortunately, these models have been based solely on observable attack surfaces present in the SDN architecture. So far, no effort was made by the community to test the validity of prevalent SDN security models against real attacks. I took the opportunity to test the validity of 2 popular SDN specific security models against the 20 attacks surveyed in this paper. As will be discussed further in Section VI-C, context specific security models like those for SDN are not able to explain some of the attacks that prey on attack surfaces that are less visible.

II. Background

Before proceeding further, it is important for the reader to get familiar with software defined networks. This section will serve as a primer and will explain some of the common terms and concepts that are referenced in subsequent sections of this paper.

The term SDN is often used both to refer to Software Defined Networking as well as a Software Defined Network depending on the context. I would like to make a note to the reader that for the remainder of this paper, the usage of SDN[s] will refer to Software Defined Network[s].

A. Networking Element

The discussion in this paper deals with different kinds of networking devices which may be Routers, Switches, Load Balancers, Firewalls etc. The term networking element will be used as a general identifier for all such networking devices. It is important to note that a server or virtal machine (VM) is not a networking element since these are usually endpoints in a network. As such a server or VM may be called a host, end-host or endpoint.

B. Control Plane

The Control Plane is the cognitive center which is concerned with network-wide logic and is agnostic about the low level details involved in a networking element's interaction with incoming data packets. The nature of the interaction may be to forward, modify or even discard said packet. The control plane will formulate some rules based on its interpretation of the network state which may either be sensed via some protocol (BGP[2] [11], STP[3] [12]) or as instructed by an administrator. These rules are then pushed into a Forwarding Table or Look-up-Table which is also accessible to the data plane. The control plane in a conventional network is distributed with each networking element having it's own control plane. In contrast, the control plane in an SDN is centralized with all networking elements sharing the same control plane.

C. Data Plane

A networking element's Data Plane interacts with data packets. It parses the header fields of incoming packets and takes actions based on rules present in the Forwarding Table. The data plane is incapable of taking action in the absence of these rules. The burden is on the control plane to ensure that the forwarding tables are always populated with rules that are accurate and up-to-date.

D. Forwarding Table

This is a table on every networking element containing rules that specify how the element must deal with data packets. The rules in these tables specify *Match-conditions* based on one or more packet headers and an associated *Action*. For every incoming data packet, the networking element will parse the packet headers and look for matching rules in the forwarding table. If a matching rule is found, the element will then take the specified action on the packet. If a matching rule is not found, in the case of SDN, the element will forward this packet to the network controller via a packet-in OpenFlow message. The actions that an element can take will depend on the type of device that element is. In general, the actions may be *Forward*, *Drop* or *Modify then Forward*.

Note that a lot of literature on SDN may use the term flow tables and flow rules in place of forwarding table and forwarding rules. They mean one and the same.

E. Conventional Network

In a conventional network, each networking element will have its own control plane and data plane. All networking elements in this network will have its own OS and will run its own instances of networking protocols. Networking elements then interact with each-other to arrive at a consensus about the state of the network as a whole. ***Fig. 1*** gives a simple representation of this scenario in a conventional network comprising only routers. It is important to note that consensus in this context is still local to each networking element as it is based on information received by the control plane of one element from that of other elements.

There are some major problems that arise in a conventional network

(1) Protocols have to be carefully and ingeniously designed so that they are quick to converge and yet also dynamic enough to automatically re-adjust in response to changes in the network. Not many protocols are really able to strike a balance in this regard. For instance, in the context of routing, OSPF[4] and BGP are two contrasting protocols that prove this very point. The former is accurate but unstable in large networks while the latter scales really well but requires careful configurations to ensure accuracy.

[2]BGP: Border Gateway Protocol

[3]STP: Spanning Tree Protocol

[4]OSPF: Open Shortest Path First

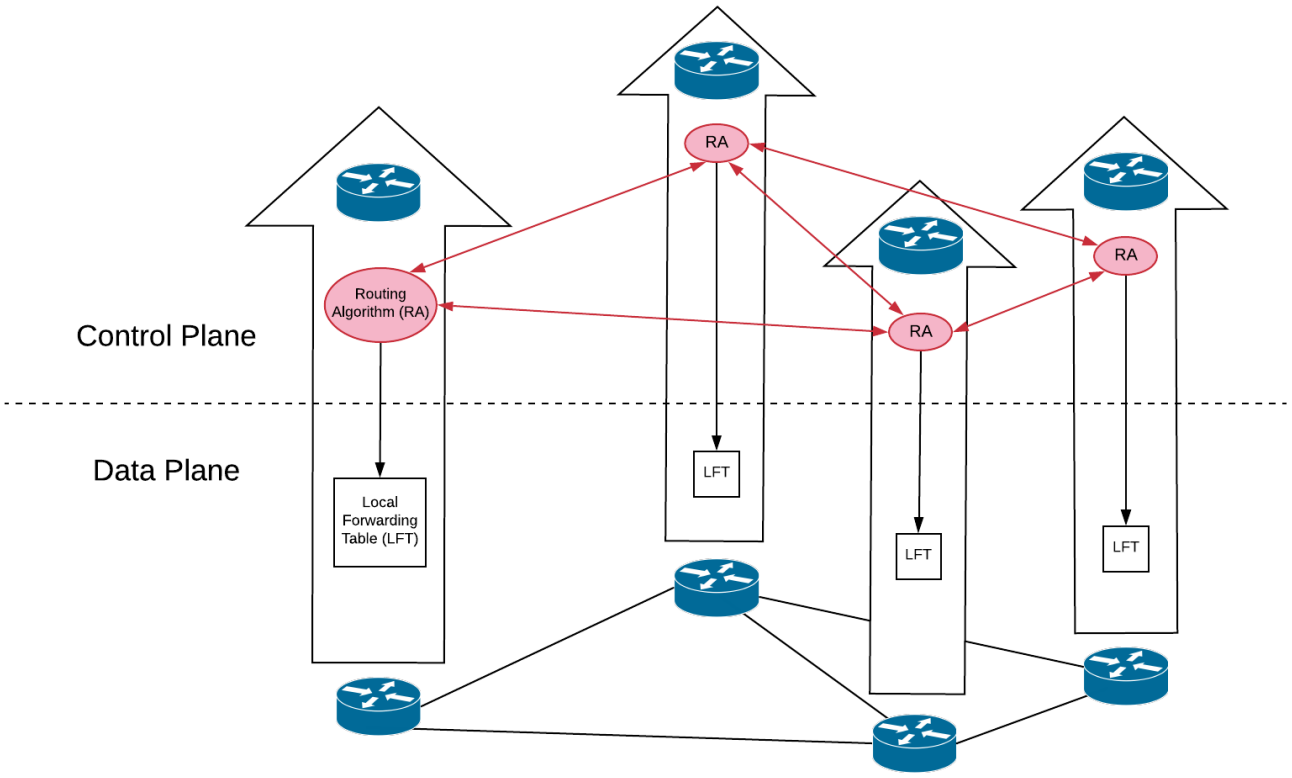

Fig. 1. A conventional network consisting of four routers. Each router runs its own instance of a routing algorithm/protocol. Adjacent devices exchange protocol related messages to first get to a consensus about the topology. Following this, each router's control plane will determine forwarding rules that are then written into the local forwarding table.

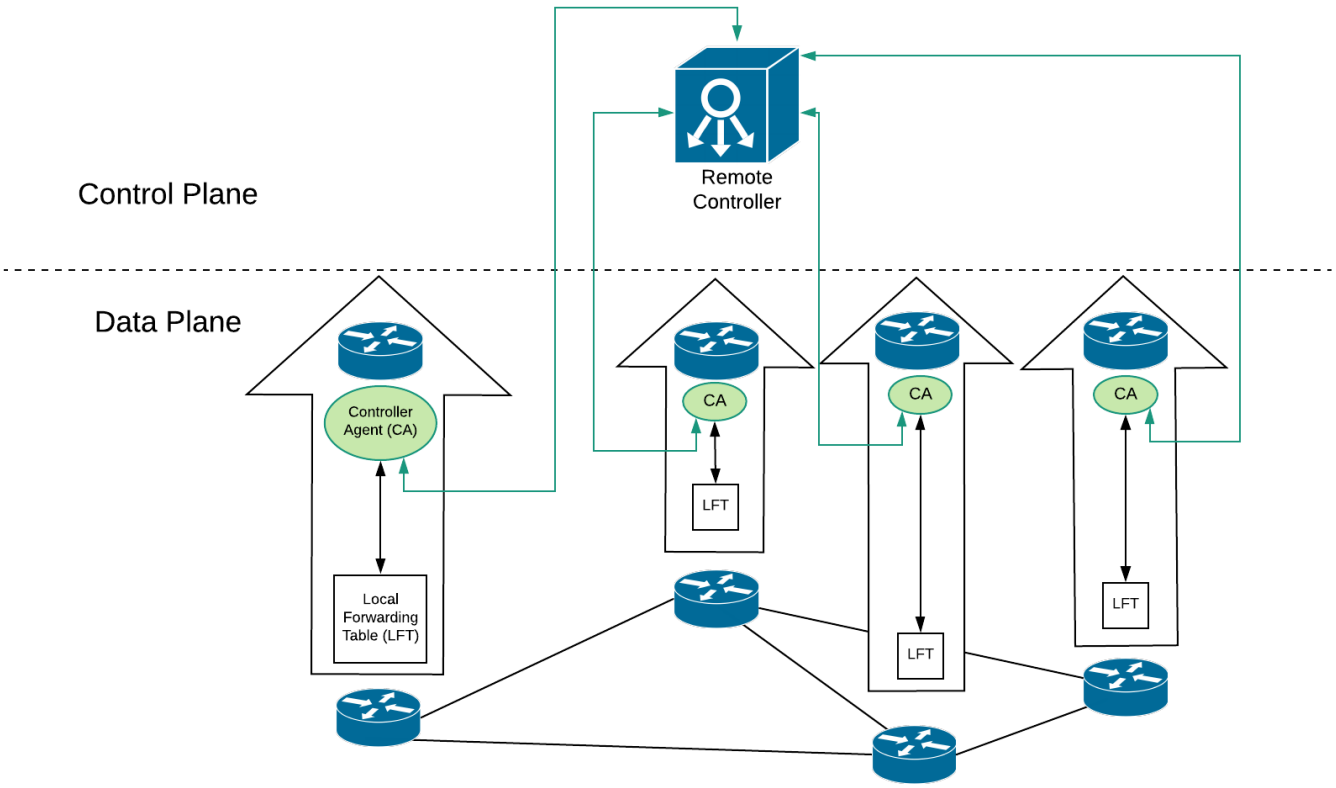

Fig. 2. The network from ***Fig. 1*** has been converted into a software defined network. With a centralized control plane, each router no longer needs to run it's own routing algorithm. The controller communicates with routers via the Controller Agent (CA) and will be aware of the complete topology. It will then run a centralized routing algorithm and push relevant forwarding rules to each router.

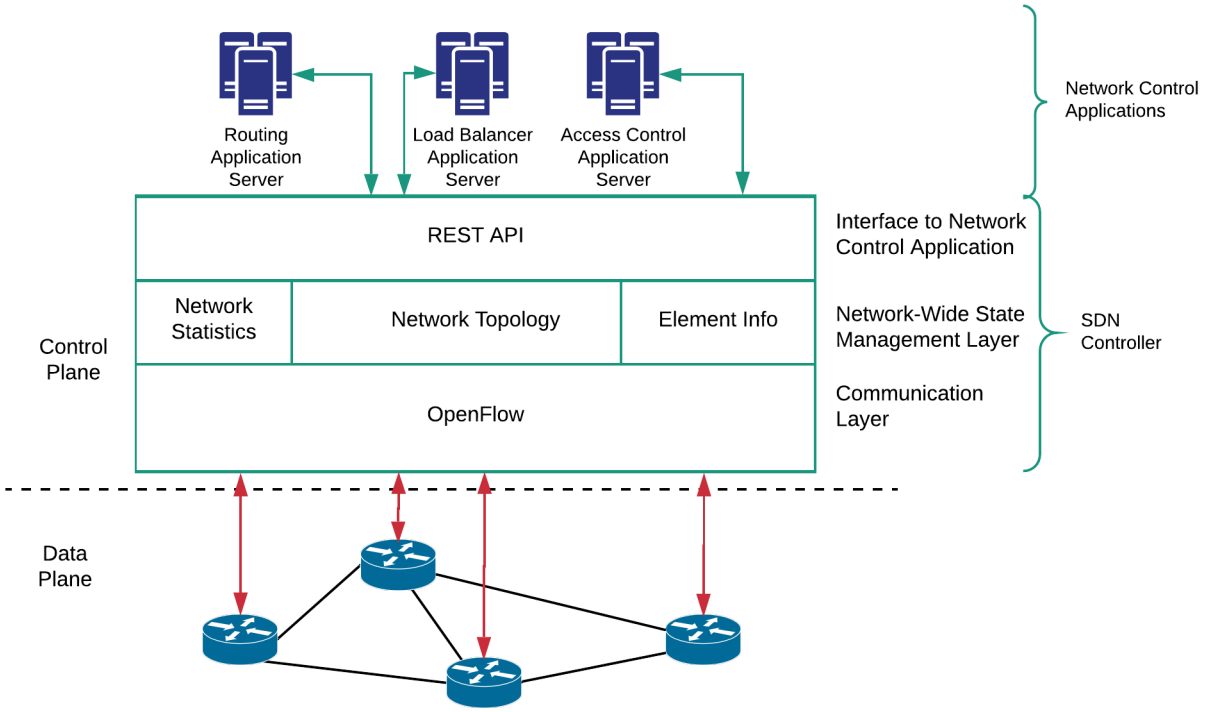

Fig. 3. The Complete SDN Architecture

(2) Since each networking element is an independent entity, it needs to be configured and managed separately. In the context of large networks like Data Centers or Internet Service Providers this translates to higher Operational Expenses (OpEx) since there is a need for more manpower.

(3) Each networking element has limited processing power and memory which is usually only sufficient for running its own OS an instances of protocols. This prevents an administrator from running custom applications or scripts to help automate device management tasks.

F. Software Defined Network (SDN)

SDNs have been conceived to address the shortcomings of conventional networks. The idea involves decoupling, in a sense, the control and data plane of each networking element. Each networking element will still maintain its own data plane but will share a centrally located control plane (usually implemented as a controller). ***Fig. 2*** gives a simple representation of this idea wherein the same network from ***Fig. 1*** has been modified to make it an SDN. The *Controller Agent (CA)* in the diagram is a generic construct that can be thought of as a logical interface on the networking element used to communicate with the controller. The centralized control plane is made up of the SDN Controller and Network Control Applications with the controller serving as an intermediary between the network control applications and the networking elements.

One could argue that a central point of control could also translate to a central point of failure. This is a valid point and is usually addressed through the presence of redundant controllers either as a backup or by having controllers operate in clusters. Suffice to say that the problem of controller redundancy is a separate field of study within SDN research.

Fig. 3 gives a complete picture of what the SDN architecture actually looks like. All terms are explained in the following subsections.

G. SDN Controller

The SDN controller has a couple of roles. The primary role is to serve as an intermediary between network control applications and network elements. Controllers are expected to have three layers. Going from bottom to top they are

(1) ***Communication Layer*** : This layer is used to facilitate bi-directional communication between the controller and network elements. OpenFlow is an example of a communication protocol that functions in this layer. The controller uses OpenFlow to make modifications to a networking element's forwarding tables.

(2) ***Network-Wide State Management Layer*** : This is the middle layer in the SDN hierarchy. This layer is responsible for maintaining an updated record of the current state of all networking elements that the SDN controller controls. In this context, state refers to up-to-date information about link status, counter values, statistics, forwarding table entries and any

other data that the network control applications may require to make decisions.

(3) ***Interface to Network Control Applications*** : This interface is implemented as an API[5] like REST[6] that network control applications use to read/write state information present in the state management layer below. This layer is also expected to provide a means for applications to get notified whenever there are changes in the state management layer.

Fig. 3 provides a cross section of the SDN controller to give a better perspective of how the different layers look.

H. Network Control Applications

Applications may be housed in one or more servers. They are responsible for sensing the network (via information present in the SDN controller's state management layer) and then computing appropriate forwarding rules for each networking element. The SDN controller will then update the forwarding tables on each networking element using the communication layer protocol (OpenFlow).

I. OpenFlow Protocol

OpenFlow is a communication protocol that runs in an SDN controller's communication layer. It is used for bi-directional communication between the controller and networking element. Communication is via TCP[7] port 6653 and encryption (TLS[8]) is optional. There are two classes of OpenFlow messages that are important.

(1) ***Controller-to-element Messages*** :

(a) *Configuration* messages to query configuration parameters from the element.

(b) *Modify-state* messages to add/remove rules in an element's forwarding table.

(c) *Read-state* messages to retrieve statistics and counter information from elements.

(d) *Send-Packet* messages to force a specific packet out of a specified port of an element.

(2) ***Element-to-Controller Messages*** :

(a) *Flow-removed* messages to inform the controller that an entry has been deleted from the forwarding table.

(b) *Port-status* messages to inform the controller about a change in status of a specific port.

(c) *Packet-in* messages to send a packet to a controller for additional processing.

III. Related Work

The first ever analysis in relation to the security of an SDN was [13] wherein Kreutz *et al.* put forward seven attack vectors that adversaries may try to exploit. They based their argument on the fact that a software defined network is seen as a honeypot because: (1) The network is controllable wholly through software. (2) Controllers are the single point of network intelligence.

[5]API: Application Programming Interface
[6]REST: Representational State Transfer
[7]TCP: Transmission Control Protocol
[8]TLS: Transport Layer Security

Zerkane *et al.* [14] carried out the most substantial work in recent years wherein they developed a generic meta-model of the SDN architecture based on an in-depth analysis that revealed 113 possible attack surfaces. They did not however test the validity of their security model against real attacks. In the years since, other researchers have proposed novel attacks like [15] and [16] that do not fit existing SDN security models.

In [17] the authors identify security challenges at various layers of the SDN architecture. They created a security model that was still derivative of Kreutz's model and nearly not as detailed as Zerkane's model. They also reviewed other research carried out in relation to data security in SDN at the time. Other works such as [18] [19] [20] are usually focused on analyses of OpenFlow or one of the many commercial controllers that implement OpenFlow.

There have been some efforts to collect and review specific attacks like Denial of Service against certain kinds of commercial controllers [21]. My search of existing publications did not reveal any prior effort to collect and review a variety of different attack types. Therefore, to my knowledge, this is the first paper to analyze a variety of different attacks proposed in recent years and derive insights both about the attacks and the adversaries.

Also, while aforementioned efforts have helped formulate multiple SDN specific security models, this is the first paper to evaluate the validity of existing security models against real attacks.

IV. Methodology

Levy and Ellis, in [22], detailed an efficient approach for carrying out literature reviews in information system research. I have used search techniques from the aforementioned work in order to find and identify relevant publications for this study. The realm of search included the databases of IEEE, Springer and ACM which have a track record for quality publication that is peer-reviewed. Some of the literature referenced in this paper were found through Google Scholar. In addition, I have also referenced documents published by the Internet Research Task Force (IRTF) and the Internet Engineering Task Force (IETF).

In total, 20 publications about attacks proposed against SDNs were identified: [15] [16] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40]. These were then analyzed to answer the following questions,

(1) Can attacks be classified in a unique way based on some characteristic of the adversary?

(2) What does the profile of an adversary look like? Are there any commonalities we can infer to design better defenses against attacks?

(3) How well do known SDN security models help predict these attacks? Are these SDN specific security models sufficient?

A. With Respect to the Attacks and the Adversary

The following set of adversarial characteristics were first defined.

(1) ***Detectability*** : Whether the adversary could be detected at any point prior-to or during the attack.

(2) ***Pre-Requisites*** : Includes the adversary's motive as well as their prior knowledge of SDN related source code.

(3) ***Location of Adversary Controlled Server/VM*** : The adversary will typically need to have a device within the target SDN to initiate the attack. The location of these devices are useful to know when planning defenses.

(4) ***Initial Capabilities*** : Some of the attacks require the adversary to have certain initial capabilities including packet manipulation or control over one or more SDN components.

Using these characteristics, adversary models behind each of the attacks were then scrutinized to determine matching characteristics if any. Following this, an adversarial profile was created.

B. With Respect to Security Models

Current SDN security models have been attempts at identifying attack surfaces, within the SDN architecture, that an adversary may target to mount an attack. Unfortunately, security models are often called threat models in SDN literature. This can be confusing since a threat model has a different meaning in the context of information security. The usage of model or security model in the remainder of this paper refers to the SDN security model as defined in this section.

For the purpose of the current study, three reference security models were first chosen.

(A) ***Kreutz Model*** : The first reference model was the original SDN security model proposed by Kreutz *et al.* [13] based on 7 attack surfaces. The motive to choose the *Kreutz Model* was borne out of the fact that this was the very first security model proposed for SDN and that most subsequent models like [17] have been derived from this.

(B) ***Zerkane Model*** : The second reference model had to be the in-depth meta model proposed by Zerkane *et al.* [14] which, with its 113 identified attack surfaces, is still to this day the most detailed SDN security model in existence.

(C) ***STRIDE Model*** : Finally, for the third model, I decided to pick the STRIDE model. This model was not specifically designed for SDN but has been popular in information security. The STRIDE model [41] was initially developed by Kohnfelder and Garg for internal use at Microsoft. Later, it garnered more popularity in the industry for being simple and thorough. STRIDE is an acronym for *Spoofing*, *Tampering*, *Repudiation*, *Information disclosure*, *Denial-of-service* and *Elevation of privilege*. This model was picked specifically to understand whether a model built for general software security would still be sufficient even in the context of SDN.

Next, each of the 20 attacks were analyzed to understand the exact vulnerability or attack surface the adversary would need to exploit for said attack to be successful. After this, I proceeded to map each attack to the reference security models. A mapping was deemed successful if the reference model acknowledged the possibility of such an attack.

V. Results

The results of the analyses have been summarized and tabulated. ***TABLE I*** indicates the adversarial profile behind each of the 20 attacks. ***TABLE II*** shows the mapping of attacks to the reference security models. The results are interpreted and discussed in the next section.

VI. Discussion

A. The Attacks

Based on the adversary's motive, the attacks can be classified into one of four types.

(1) ***Man in the Middle Attacks*** : The adversary is interested in inserting himself between two communication endpoints so that he may eavesdrop or siphon traffic out for detailed analysis. The attacker can achieve this by compromising one or more network elements in the SDN [23]. Even if the attacker is unable to gain control of a network element, the attack can still be carried out by some interesting traffic manipulation. That is, the attacker can craft packets to poison the ARP[9] cache [25] of target endpoints so that they end up sending traffic to the adversary's host instead of a legitimate SDN networking element.

(2) ***Complete Control Attacks*** : Here, the adversary wants to assume complete control of the target SDN. If the attack succeeds, the adversary will have as much power over the SDN as the network administrator. The adversary has a number of way to go about this. They can exploit known vulnerabilities in SDN Controllers which could allow for remote code execution. As was shows in [36] by Ropke and Holz, rootkits can be installed on SDN controllers either through a compromised SDN application or even through specially crafted packets. Once installed, the rootkits give the adversary complete control over the controller while hiding its presence so that the defender is unaware. In [16], the authors were able to get shell access to all controllers in the SDN by injecting a malformed MPLS[10] packet containing shellcode embedded in the packet header. Similar attacks can also be launched in virtual networks located in the public cloud. The consequences of this form of attack on the public cloud are more dire as the adversary may then gain complete control of not just the SDN but also the entire cloud system [38].

Authors of [28] propose an interesting attack that could be used by an adversary to operate their own flows in the network without alerting the network administrators. This attack is novel in that it subverts the *packet-in* and *send-packet* OpenFlow messages to have controllers deliver packets directly to specific points in the network. These packets are "*teleported*" thus bypassing security appliances like firewalls or Intrusion Detection Systems (IDS).

Two other attacks are the Man At The End (MATE) [39] and the ATTAIN attack injection framework [27]. These are attack frameworks which can be used to mount different kinds of attacks on the SDN. These attacks are generic enough and

[9]ARP: Address Resolution Protocol

[10]MPLS: Multi-Protocol Label Switching

TABLE I
ADVERSARY PROFILE

	Pre-Requisites			*Location of Adversary's Server/VM*			*Adversary's Initial Capabilities*			
Attacks	*Can Adversary be Detected?*	*Adversary's Motive*	*Adversary has Analyzed Controller Source Code*	*In Target SDN*	*Within SDN Controller's Subnet*	*Connected to SDN Switch*	*Packet Manipulation*	*Controls Compromised SDN Switch[es]*	*Controls Compromised SDN Controller*	*Controls SDN Application[s]*
[15]		REC		✓			✓			
[16]	✓	CC		✓		✓	✓			
[23]	✓	MitM		✓				✓		
[24]	✓	DoS		✓		✓	✓			
[25]	✓	MitM		✓			✓			
[26]		REC		✓			✓			
[27]	✓	CC		✓	✓	✓	✓			
[28]		CC						✓		
[29]	✓	DoS		✓	✓	✓	✓			
[30]		REC		✓	✓	✓				
[31]	✓	CC								✓
[32]	✓	DoS		✓		✓	✓			
[33]	✓	DoS		✓		✓	✓			
[34]		REC		✓			✓			
[35]		REC		✓			✓			
[36]	✓	CC							✓	✓
[37]	✓	DoS	✓	✓		✓				
[38]	✓	CC		✓			✓			
[39]	✓	CC	✓	✓	✓	✓		✓	✓	✓
[40]	✓	DoS		✓		✓	✓			

MitM Man in the Middle
CC Complete Control
DoS Denial of Service
REC Reconnaissance

can be framed in any number of ways depending on the initial capabilities of the adversary. Suffice to say that the motive of the adversary is still complete control.

(3) ***Denial of Service (DoS) Attacks*** : The intention of the adversary here is to disrupt legitimate function in an SDN. Due to the nature of how SDNs work (with respect to centralized control), this kind of attack is fairly easy to design. For instance, the attacker can flood streams of packets that are randomly generated and whose destination is not within the SDN. An SDN networking element seeing such a packet will be unaware of how to deal with it. The element will then, through an OpenFlow *packet-in* message deliver the packet to the SDN controller and await further instructions. The communication channel between the element and the controller is significantly smaller compared to physical link bandwidth. Thus, by creating enough random traffic, the attacker can choke the communication channel between the element and controller thereby rendering the SDN useless. In [24] the authors demonstrate this phenomenon by flooding an SDN switch with random UDP[11] traffic. Another form of this attack is possible when an SDN is designed in a way that the control traffic between a network element and a controller are in-band and move through the same paths as regular data traffic. All the adversary needs to do is first identify shared links and then proceed to congest them by flooding traffic [29]. Smyth and his colleagues designed yet another variation of this in [40], called an Event Flooding Attack (EFA) which could disrupt multiple controllers in a distributed SDN.

Adversaries can also deny service to end hosts by tampering with the routing paths within the SDN. They do this by distorting the controller's perspective of how the topology looks like. For instance, the attacker can send spoofed link discovery packets to SDN elements resulting in confusion about the network topology. In particular, an adversary may trick the controller into thinking there are new connections between several SDN switches which were otherwise not connected. A controller upon seeing these new paths will then re-compute routing paths and install incorrect forwarding rules in order to try and send traffic through these fake "*new*" links [32]. To a similar effect, attackers can also distort the controller's view of where end hosts reside in the SDN [33].

Xu *et al.* developed a more sophisticated and novel DoS attack [37] wherein the attacker, who is familiar with the controller's source code, is able to exploit race conditions in the controller's operating system to disrupt critical functions such as Network Address Translation (NAT) and Dynamic Host Configuration Protocol (DHCP).

(4) ***Reconnaissance Attacks*** : These are, by far, the most ingenious kinds of attacks. The adversary does not intend to disrupt the function of an SDN. Rather, the intention is to gather as much information as possible about the target SDN. The information that the adversary gains may then be analyzed and interpreted to create a blueprint of the target network which can be used to plan or develop other attacks. Alternatively, the information can be commoditized and sold to other attackers or to a competing network operator.

In a reconnaissance attack, the adversary performs one or more actions that help them detect information leaked by the SDN. What is interesting here is that a reconnaissance attack does not exploit any particular vulnerability or attack surface. That is, the actions of the adversary are not unique

[11]UDP: User Datagram Protocol

TABLE II
MAPPING ATTACKS TO THREE SECURITY MODELS

		Security Model		
Attacks	**Motive**	*Kreutz Model*	*Zerkane Model*	*STRIDE Model*
[15]	REC			✓(I)
[16]	CC	✓	✓	✓(E)
[23]	MitM	✓	✓	✓(T)
[24]	DoS	✓	✓	✓(D)
[25]	MitM	✓	✓	✓(S)
[26]	REC			✓(I)
[27]	CC	✓	✓	✓(T,D)
[28]	CC		✓	✓(T)
[29]	DoS	✓	✓	✓(D)
[30]	REC			✓(I)
[31]	CC		✓	✓(T)
[32]	DoS		✓	✓(S)
[33]	DoS		✓	✓(S,T)
[34]	REC			✓(I)
[35]	REC			✓(I)
[36]	CC		✓	✓(E)
[37]	DoS	✓	✓	✓(D)
[38]	CC	✓	✓	✓(E)
[39]	CC	✓	✓	✓(T,D)
[40]	DoS	✓	✓	✓(D)

STRIDE Acronym for **S**poofing, **T**ampering, **R**epudiation, **I**nformation Disclosure, **D**enial of Service, **E**levation of Privilege

MitM Man in the Middle
CC Complete Control
DoS Denial of Service
REC Reconnaissance

or otherwise distinguishable from that of a normal host in the network. The attacker does not directly probe or scan any SDN switch or controller. Rather, in most of the attacks, they send a specific stream of harmless traffic. For instance, Shin and Gu [26] demonstrated how response times in a network could be used to make a statistical inference about whether a target network was a conventional network or an SDN. Conti *et al.* took this a step further to develop a Know Your Enemy (KYE) attack [35] which used carefully crafted probing traffic to determine overall network control logic in an SDN. As a result, reconnaissance attacks are in general hard to detect.

Researchers have also developed reconnaissance attacks aimed at gaining information about the forwarding tables within SDN elements. In [34], the authors developed an algorithm which could be used to identify when a flow table exceeded capacity and overflowed. They were then able to test the attack and accurately determine the flow table capacity of SDN switches in the target SDN. A more sophisticated attack involved the use of a suite of methods involving several different types of probing traffic by which the author was able to make inferences and accurately reconstruct the forwarding rules present in an SDN switch's forwarding table [15].

Reconnaissance attacks are not limited to targeting SDN controllers and SDN elements alone. They may also target network control applications. Cao *et al.* [30] were able to make use of deep machine learning to analyze SSL[12] encrypted traffic between SDN controllers and elements. This was then used to determine the kind of network control applications that were being used in the target SDN. What is interesting here is that the authors demonstrated that even though the content of the traffic was encrypted, the packets themselves followed certain patterns that a deep neural network was able to glean.

I would like to make a note to the reader that the phenomenon of a reconnaissance attack is very new and was made more severe to me owing to the sophistication of the attacks and the detail of the information that an adversary could derive from an SDN. To my knowledge, there has not yet been any prior paper or publication that was focused on studying reconnaissance attacks against SDN or about the detectabilty of the adversary in case of reconnaissance attacks.

B. The Adversary

(1) ***Detectability*** : Detectability allows for a defender to detect adversaries based on their actions. By being able to detect an adversary, the defender has some chance at taking countermeasures against any given attack. It is possible to detect the presence of an adversary in an SDN prior to or during an attack as long as the attack isn't a reconnaissance attack. This is because, in a reconnaissance attack, the adversary does not take actions that are otherwise different from actions taken by legitimate participants of that network. In the case of all other attack types, the adversary will take actions that cannot go unnoticed. This may be in the form of a sudden spike of randomly generated traffic (in case of a DoS attack like [24] [37]) or a network element that has gone rogue [16] and can no longer be controlled by the administrator.

To defend against reconnaissance attacks, a defender now has to implement proactive measures in an effort to either minimize the information leak or scramble the information getting leaked thereby making it useless to the adversary. For instance, to defend against reconnaissance attacks that attempt to derive information about the forwarding tables the authors of [15] discuss some novel strategies like repeated address randomization and multi-layered encapsulation. While they are effective, it should be noted that existing strategies to thwart reconnaissance attacks impose a significant overhead on SDN controllers and elements. There is a good opportunity for the research community to come up with efficient mechanisms to defend against reconnaissance attacks.

(2) ***Pre-Requisites*** : There is an implicit assumption made that all adversaries have a working knowledge about SDN. Adversaries may still be interested in analyzing code and finding vulnerabilities. For instance, in [37] the adversary

[12]SSL: Secure Socket Layer

has analyzed and identified race conditions which are then exploited to mount a DoS attack.

Most of the code used in SDN controllers are open sourced and publicly available. While this does let an adversary access the code, it also allows for the community to analyze, report and patch any vulnerabilities. Researchers working on [16] identified two such vulnerabilities in OpenvSwitchd's packet parser during the course of their study and reported this (CVE-2016-10377 [42]) to the OpenStack community. The vulnerability was addressed in the subsequent version of OpenvSwitchd.

On the whole, keeping the code open source has helped ensure that (1) good code is written and that (2) anyone can perform analysis and suggest corrections. The scenario may change once more vendors start to develop closed proprietary source code for their SDN products.

(3) ***Location of Adversary Controlled Server/VM*** : As indicated in ***TABLE I***, irrespective of the motive and type of attack, nearly all attacks require the adversary to have a server/vm in the target SDN to initiate the attack. The exception being for attacks like [36] where the attack is initiated from a malicious application on the SDN controller. This means that for 17 out of the 20 attacks that were surveyed, a defender could have thwarted the adversary by having stricter control over end hosts in their SDN. While the previous statement may be sound, the situation is a little more nuanced and can vary for different kinds of SDN deployments.

If the SDN is that of an application or website or service, the chances are that all servers are completely under the control of the network administrator. In such a scenario it is possible for the admin to create strict policies in relation to the kind of traffic that the servers are allowed to transmit. Moreover, the admins will also have a say in the deployment of software ensuring that servers are always running secure and up-to-date code. I do concede that what I just described is an idealized scenario and that even the most cautions admin can be guilty of oversight. What I meant is that with strict administrative controls in place, adversaries often have little to no chance of gaining complete control of a server/vm in the target SDN thereby significantly reducing the possibility of an attack.

On the other hand, if the SDN were that of a service provider (SP) the circumstances would be different. Typical SP networks allow for end customers to lease servers or lease resources to run their own VMs. Restrictions are far less than compared to the previous scenario and the administrators don't really have direct control of what kind of software that their clients choose to run. All it takes then is a careless user, running vulnerable code, whose device gets taken control of by an adversary. Once the adversary is in the network, he might even assume control of the entire service provider infrastructure as was demonstrated in [38].

Another point of consideration was the specific area of the SDN wherein the adversary's server/vm was located. Having their device connected to an SDN switch is not strange since servers are typically connected directly to a Top-of-Rack (TOR) switch. It is common practice for admins to restrict or to indiscriminately drop protocol related packets (routing/switching related) received on a switch port connected to a server since end hosts are not expected to be running specific protocols such as STP[13]. Unfortunately, such restrictions can't deter attacks like [25] or [32] which make use of spoofed ARP[14] and spoofed LLDP[15]; both of which are protocol packets that servers are expected to generate.

Finally, it is generally considered as bad design practice for management traffic [16] and production traffic [17] to co-exist in the same physical or logical network. Thus, having the adversary or any server (which is not under admin control) located in the same subnet as that of the SDN controller is a bad idea as is demonstrated by the authors of [29] and [30]. Thus, following best practices in network design can help prevent some obvious attacks.

Overall, it is easy enough to appreciate the fact that keeping the adversary out of the network or at least limiting their capabilities within the SDN will go a long way in hardening defenses. But as we have seen, there is a need to design strategies and policies that are suitable for specific applications of SDNs since no two networks are identical.

(4) ***Initial Capabilities*** : Most attacks (13 out of 20) require the adversary to have some kind of packet manipulation capability. Packet manipulation here refers to the ability to craft and transmit customized data packets. This capability is essential for adversaries interested in conducting DoS attacks either to overload control plane communication [24] or to distort the controller's view of the network topology [32] [33]. In both cases, the adversary can be thwarted by using intelligent firewalls. Reconnaissance attacks also make use of crafted packets but these are usually undetected by firewalls since the adversary is careful and (1) generates a low volume of packets with (2) patterns that are not distinguishable from legitimate host traffic.

Some other attacks assume that adversary has control over one or more SDN components prior to the attack. It should be noted that an adversary that has control over an SDN controller or SDN application has enough power to take down the entire network. The worst sort of scenario arises when the adversary's software is intelligent enough to mask its presence [36] making it hard for the defender to track.

As a defender, the objective should be to limit the capabilities of the adversary. To that end, by enforcing reasonable access control mechanisms, keeping all SDN software up-to-date and running code from trusted repositories it should be easy enough to prevent the adversary from hijacking one or more SDN components. The same cannot be done when it comes to restricting the packet manipulation capabilities of

[13]STP: Spanning Tree Protocol

[14]ARP: Address Resolution Protocol

[15]LLDP: Link Layer Discovery Protocol

[16]Management traffic refers to traffic related to administrative functions. Communication between SDN switches and their controllers fall in this criteria. Usually, the management network is physically or at least logically separated from the production network.

[17]Production traffic refers to all data traffic and some control traffic that cannot be carried over the management network.

the adversary. As discussed before, the network administrator may be limited in terms of the policies that can be enforced on the end user's capabilities depending on circumstances of that particular SDN.

C. The Security Models

As can be seen from ***TABLE II*** many attacks could not be mapped to SDN specific security models while all attacks could successfully be mapped to the STRIDE model.

The *Kreutz model* was very simplistic and considered only seven possible attack surfaces that an adversary could use to mount attacks against an SDN. While this model was significant at the time of its publication, it is not without some deficiencies. For instance, the model fails to explain reconnaissance attacks that do not exploit any observable attack surface. Another flaw in the model is that it does not consider the possibility of vulnerabilities in software [28] nor that network control applications could be malicious [36].

The *Zerkane model* was very thorough since the authors created it through a rigorous analysis of the SDN architecture by first identifying every single component in an SDN and subsequently every possible attack surface on each component. By paying attention to every possible attack surface the *Zerkane model* also considers software vulnerabilities and malicious applications. This model is no doubt an improvement from the *Kreutz model* but is still completely based on observable attack surfaces. As a result, it fails to explain reconnaissance attacks which do not exploit any observable attack surface.

The *STRIDE model* is the simplest out of the three reference models but still encompasses all possible threats that have been studied and developed as part of research into information security. As a result, the STRIDE model is able to explain all attacks including the reconnaissance attack which aims to capture information leaked by the SDN. Moreover, as ***TABLE II*** indicates, there are a few overlaps too with some of the attacks like [27] falling into more than one criteria that STRIDE describes. STRIDE's success here is is based on the fact that it is based on attack type and not based on attack surface.

There is no doubt that SDN security models are relevant and are useful in predicting attacks and creating defenses. However, the findings of this study highlight the shortcomings of defining security models merely based on observable attack surfaces. Both the *Kreutz model* and the *Zerkane model* were built from the ground-up by considering the very special SDN use case. SDN is not special or very different from any other information system. All threats affecting typical information systems are also threats to SDN. Moving forward, it is important for the SDN community to to develop security models holistically and not just based on attack surfaces. The security of SDN should be analyzed with the same scrutiny and rigor with which one would analyze any other information system.

VII. Conclusion

Over the course of this survey I have highlighted several points about the security of SDN that require focus.

(1) SDNs leak information by virtue of its architecture and adversaries have begun to find ways to access this by means of several novel reconnaissance attacks. There is a need to address the information leaks by making modifications to the architecture, developing efficient proactive countermeasures or by improving the detectability of the this kind of attack.

(2) An adversarial profile helps compare and contrast the various capabilities of the adversary thereby helping a defender draw sufficient insights and plan better defenses. That said, in order to keep up to date the profile needs to be periodically refreshed both by analyzing newer attacks as well as by collecting even more parameters about the attacks and adversaries.

(3) Current SDN security models have some limitations that need to be addressed. The security of SDNs need to be scrutinized the same way as any other information system.

In addition to these points, all prior work has assumed a general SDN and has so far been agnostic about the application or use-case of the network. It will be interesting to assess whether threat considerations and adversary models will change depending on the kind of application that an SDN is used for. For instance, a network serving as a data center interconnect [43] will be different from one functioning as a Content Distribution Network (CDN) [44] or one which serves as a Vehicular Ad-hoc Network (VANET) [45].

SDN is rapidly evolving and is gaining prominence by way of its increased usage in publicly accessible networks. While security analyses are carried out from time to time, it is absolutely necessary for researchers to keep challenging existing perceptions and ideas about securing SDN. Such efforts will no doubt help suitably harden SDN before it is used in more mainstream and critical computer networks.

References

[1] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: An intellectual history of programmable networks," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, p. 87–98, Apr. 2014. [Online]. Available: https://doi.org/10.1145/2602204.2602219

[2] D. S. Alexander, W. A. Arbaugh, M. W. Hicks, P. Kakkar, A. D. Keromytis, J. T. Moore, C. A. Gunter, S. M. Nettles, and J. M. Smith, "The switchware active network architecture," *IEEE Network*, vol. 12, no. 3, pp. 29–36, May 1998.

[3] T. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo, "The softrouter architecture," in *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking*, vol. 2004. Citeseer, 2004.

[4] A. Farrel, J. Vasseur, and J. Ash, "Rfc 4655: A path computation element (pce)-based architecture," *IETF, August*, 2006.

[5] 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.

[6] K. Greene, "Tr10: Software-defined networking," *Technology Review (MIT)*, 2009.

[7] E. Haleplidis, K. Pentikousis, S. Denazis, J. Salim, D. Meyer, and O. Koufopavlou, "Rfc 7426: Software-defined networking (sdn): layers and architecture terminology," *Internet Research Task Force (IRTF)*, 2015.

[8] 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, no. 4. ACM, 2013, pp. 3–14.

[9] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega *et al.*, "Network slicing to enable scalability and flexibility in 5g mobile networks," *IEEE Communications magazine*, vol. 55, no. 5, pp. 72–79, 2017.

[10] E. Mohyeldin, "Minimum technical performance requirements for imt-2020 radio interface (s)," in *ITU-R Workshop on IMT-2020 Terrestrial Radio Interfaces*, 2016, pp. 1–12.

[11] Y. REKHTER, "Rfc 4271 : A border gateway protocol 4 (bgp-4)," *http://www.ietf.org/rfc/rfc4271.txt*, January 2006. [Online]. Available: https://tools.ietf.org/html/rfc4271

[12] "Ieee standard for local and metropolitan area networks: Media access control (mac) bridges," *IEEE Std 802.1D-2004 (Revision of IEEE Std 802.1D-1998)*, pp. 1–281, June 2004.

[13] D. Kreutz, F. M. 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*, ser. HotSDN '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 55–60. [Online]. Available: https://doi.org/10.1145/2491185.2491199

[14] S. Zerkane, D. Espes, P. Le Parc, and F. Cuppens, "Vulnerability analysis of software defined networking," in *Foundations and Practice of Security*, F. Cuppens, L. Wang, N. Cuppens-Boulahia, N. Tawbi, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2017, pp. 97–116.

[15] S. Achleitner, T. La Porta, T. Jaeger, and P. McDaniel, "Adversarial network forensics in software defined networking," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 8–20. [Online]. Available: https://doi.org/10.1145/3050220.3050223

[16] K. Thimmaraju, B. Shastry, T. Fiebig, F. Hetzelt, J.-P. Seifert, A. Feldmann, and S. Schmid, "Taking control of sdn-based cloud systems via the data plane," in *Proceedings of the Symposium on SDN Research*. ACM, 2018, p. 1.

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

[18] R. Klöti, V. Kotronis, and P. Smith, "Openflow: A security analysis," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, Oct 2013, pp. 1–6.

[19] K. Benton, L. J. Camp, and C. Small, "Openflow vulnerability assessment," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13. New York, NY, USA: ACM, 2013, pp. 151–152. [Online]. Available: http://doi.acm.org/10.1145/2491185.2491222

[20] R. K. Arbettu, R. Khondoker, K. Bayarou, and F. Weber, "Security analysis of opendaylight, onos, rosemary and ryu sdn controllers," in *2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*, Sep. 2016, pp. 37–44.

[21] H. Polat and O. Polat, "The effects of dos attacks on odl and pox sdn controllers," in *2017 8th International Conference on Information Technology (ICIT)*, May 2017, pp. 554–558.

[22] Y. Levy and T. J. Ellis, "A systems approach to conduct an effective literature review in support of information systems research." *Informing Science*, vol. 9, 2006.

[23] M. Antikainen, T. Aura, and M. Särelä, "Spook in your network: Attacking an sdn with a compromised openflow switch," in *Nordic Conference on Secure IT Systems*. Springer, 2014, pp. 229–244.

[24] H. Wei, Y. Tung, and C. Yu, "Counteracting udp flooding attacks in sdn," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, June 2016, pp. 367–371.

[25] M. Brooks and B. Yang, "A man-in-the-middle attack against opendaylight sdn controller," in *Proceedings of the 4th Annual ACM Conference on Research in Information Technology*, ser. RIIT '15. New York, NY, USA: ACM, 2015, pp. 45–49. [Online]. Available: http://doi.acm.org/10.1145/2808062.2808073

[26] S. Shin and G. Gu, "Attacking software-defined networks: A first feasibility study," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13. New York, NY, USA: ACM, 2013, pp. 165–166. [Online]. Available: http://doi.acm.org/10.1145/2491185.2491220

[27] B. E. Ujcich, U. Thakore, and W. H. Sanders, "Attain: An attack injection framework for software-defined networking," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2017, pp. 567–578.

[28] K. Thimmaraju, L. Schiff, and S. Schmid, "Outsmarting network security with sdn teleportation," in *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, April 2017, pp. 563–578.

[29] J. Cao, Q. Li, R. Xie, K. Sun, G. Gu, M. Xu, and Y. Yang, "The crosspath attack: Disrupting the SDN control channel via shared links," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 19–36. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/cao

[30] J. Cao, Z. Yang, K. Sun, Q. Li, M. Xu, and P. Han, "Fingerprinting SDN applications via encrypted control traffic," in *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*. Chaoyang District, Beijing: USENIX Association, Sep. 2019, pp. 501–515. [Online]. Available: https://www.usenix.org/conference/raid2019/presentation/cao

[31] S. Lee, C. Yoon, and S. Shin, "The smaller, the shrewder: A simple malicious application can kill an entire sdn environment," in *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, ser. SDN-NFV Security '16. New York, NY, USA: ACM, 2016, pp. 23–28. [Online]. Available: http://doi.acm.org/10.1145/2876019.2876024

[32] Tri-Hai Nguyen and Myungsik Yoo, "Analysis of link discovery service attacks in sdn controller," in *2017 International Conference on Information Networking (ICOIN)*, Jan 2017, pp. 259–261.

[33] T. Nguyen and M. Yoo, "Attacks on host tracker in sdn controller: Investigation and prevention," in *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, Oct 2016, pp. 610–612.

[34] Y. Zhou, K. Chen, J. Zhang, J. Leng, and Y. Tang, "Exploiting the vulnerability of flow table overflow in software-defined network: Attack model, evaluation, and defense," *Security and Communication Networks*, vol. 2018, 2018.

[35] M. Conti, F. De Gaspari, and L. V. Mancini, "Know your enemy: Stealth configuration-information gathering in sdn," in *International Conference on Green, Pervasive, and Cloud Computing*. Springer, 2017, pp. 386–401.

[36] C. Röpke and T. Holz, "Sdn rootkits: Subverting network operating systems of software-defined networks," in *International Symposium on Recent Advances in Intrusion Detection*. Springer, 2015, pp. 339–356.

[37] L. Xu, J. Huang, S. Hong, J. Zhang, and G. Gu, "Attacking the brain: Races in the SDN control plane," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 451–468. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/xu-lei

[38] K. Thimmaraju, B. Shastry, T. Fiebig, F. Hetzelt, J. Seifert, A. Feldmann, and S. Schmid, "Reigns to the cloud: Compromising cloud systems via the data plane," *CoRR*, vol. abs/1610.08717, 2016. [Online]. Available: http://arxiv.org/abs/1610.08717

[39] A. E. W. Eldewahi, A. Hassan, K. Elbadawi, and B. I. A. Barry, "The analysis of mate attack in sdn based on stride model," in *Advances in Internet, Data & Web Technologies*, L. Barolli, F. Xhafa, N. Javaid, E. Spaho, and V. Kolici, Eds. Cham: Springer International Publishing, 2018, pp. 901–910.

[40] D. Smyth, D. O'Shea, V. Cionca, and S. McSweeney, "Attacking distributed software-defined networks by leveraging network state consistency," *Computer Networks*, vol. 156, pp. 9 – 19, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S138912861830940X

[41] L. Kohnfelder and P. Garg, "The threat to our products," April 1999. [Online]. Available: https://adam.shostack.org/microsoft/The-Threats-To-Our-Products.docx

[42] *Nvd.nist.gov*, 2019. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2016-2074

[43] R. Jain and R. Khondoker, "Security analysis of sdn wan applications—b4 and iwan," in *SDN and NFV Security*. Springer, 2018, pp. 111–127.

[44] N. I. Mowla, I. Doh, and K. Chae, "Multi-defense mechanism against ddos in sdn based cdni," in *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, July 2014, pp. 447–451.

[45] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, E. Cerqueira *et al.*, "Towards software-defined vanet: Architecture and services." in *Med-Hoc-Net*, 2014, pp. 103–110.