Security Enhancement in Software Defined Networking (SDN): A Threat Model

Pradeep Kumar Sharma, Dr. S.S Tyagi
Department of Computer Science and Engineering
Manav Rachna International Institute of Research and Studies, Faridabad, Haryana, India

***Abstract*—Software Defined Networking (SDN) has emerged as a technology which can replace the prevalent vendor based proprietary CLI networking devices. SDN has introduced applications based network control and provided various opportunities and challenges for research and innovation in these networks. Despite many advantages and opportunities in SDN, security is a matter of concern for developers who want to invest in SDN. In this paper we are analyzing the SDN security issues with their countermeasures. We have generalized four use cases threat model that should cover security requirements of SDN. These use cases are: (I) protect controllers from applications, (II) inter-controller protection, (III) protecting data plane or switches from controller, (IV) protecting controllers from malicious switches. We found that these SDN components are inter-related if one is secure another one is already secure. We also compared the SDN and traditional network security in terms of these four use cases and provide the insights for protection mechanism and security enhancements. A framework for the development of a SDN security application has been presented based on ryu controller. We believe that our threat model will help various researchers and developers to understand current security requirements and provide a ready reference to tackle vulnerabilities and threats in this area. Finally, we identify some open research problems and future research directions with a proposed security architecture.**

Keywords—Software defined networking (SDN); openflow; control plane; data plane; controller; programmability

I. Introduction

Traditional network (TN) devices are very powerful and provide various networking control functions in the form of routers, switches, firewall and load balancer etc. But security is always a big concern due to distributed nature of network containing various devices for various networking functions [1]. A lot of new models are being developed every year with more processing powers and updated software versions by the vendors and customer need to replace the previous hardware for getting new updated software functions. These proprietary devices are very costly and have their own way of configuration through CLI, having some specific commands and different vendors have different commands to communicate with these devices. This may results in configuration errors and various security breaches [2]. The output of these commands is as per human operator in mind and this output cannot be used further to provide programmability. Hence there is no scope for network engineers and researchers who want to scale and automate their network operations as per demands [3]. These hardware dependent systems, tightly coupled with software have failed to evolve the networking world as compare to system administration where software is independent of the hardware. In system administration, operating system is a piece of software which is not tightly coupled with hardware. We are free to install any operating system and applications on any hardware as per the requirement. As a result, system administration is evolving very fast. Today we can install many servers on a single hardware by using hypervisor, which manages several virtual machines with different host operating system. Not even hypervisor, Docker is another solution which provides high level resource utilization [4] as shown in Fig. 1 and 2 respectively.

In virtual machines concept as shown in Fig. 1, we assign dedicated processing resources and operating system to a VM image which is used by a dedicated service but Docker provides containers for hosting the specific services or applications which consumes very little resources as compare to virtual machine as shown in Fig. 2. One Docker engine can contain thousands of containers running various applications specific servers on a single operation system. On the other hand, in network administration we are still working on hardware dependent networking devices which consume a lot of processing power and time on manual configurations. There is a need to redesign the present networking architecture which can full fill the above said requirement with flexibility, programmability and automation.

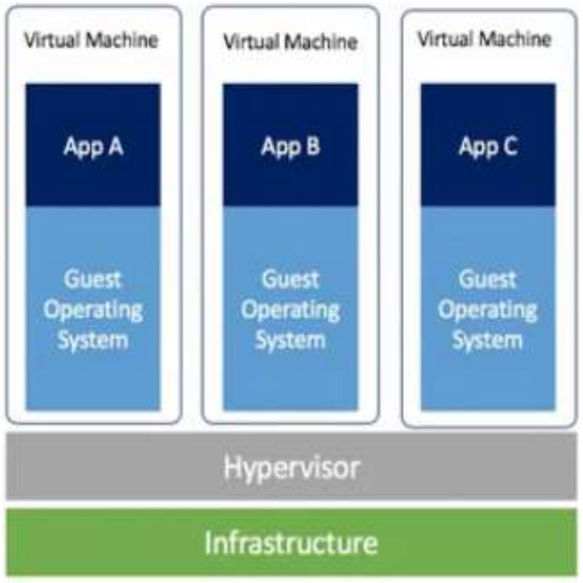


Fig. 1. Virtual Machines Hosted on Hypervisor.

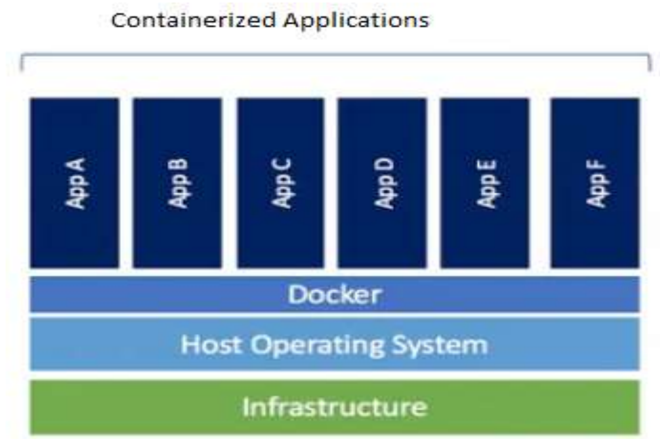


Fig. 2. Containerized Applications on Single OS through Docker.

Software Defined Networking [5] is a new concept which provides an API for configuration and decouples software logic from the devices. These devices work as simple data forwarding devices. The software or logical intelligence has been placed in a centralized controller. . The communication of forwarding devices and controller is established through a southbound API e.g. openflow [3]. All the networking functions like Routing, Security and Network monitoring etc. are done through the applications in application plane. The communication of application plane and controller is coordinated by northbound API e.g. RESTful API [6]. This provides the programmability approach and various applications can be designed a per the network demands. Network engineers can also use third party applications irrespective of hardware based solution for managing their network infrastructure. The idea of SDN is to use vendor specific hardware and we are free to choose software as per network demands irrespective of hardware. This arrangement of network functionality provides various opportunities for research and innovation in these networks. SDN is evolving and it has various advantages or traditional networks like dynamic control, programmability and a complete view of the network. As it is a new technology security solutions in SDN need to redefine and it provides various challenges and opportunities.

The rest of the contents of paper have been presented as under: Section II discusses the related works; a proposed threat model has been depicted in Section III. Comparative analysis of threats in SDN and traditional networks based on threat model has been elaborated in Section IV. Lessons learned and security enhancements by developing a security application have been discussed Section V. Section VI is dedicated to future research directions with a proposed security model. In Section VII we conclude our analysis with open research problems.

II. Related Work

Although there are several papers which provide various studies on SDN security, they do not focus on protecting SDN components from each other as SDN components are interlinked to each other and one component can attack the another component. If one component is malicious it can harm the other SDN components e.g. if application is malicious it can attack the controller and vice versa. Based on this concept we have derived four use cases to analyze the SDN security requirements and their counter measures. Also we have applied our threat model in traditional networks (TN) to analyze how these use cases are tackled in TN. We also provide a comparative analysis to find out the real threats in SDN and their possible resolutions.

In [7] Ali et al undertook a survey of related work in the area of SDN security. They presented programmable networks as an opportunity to improve protection in enterprise network through all the logical control at a centralized place. Real time policy enforcement and flexibility are presented as key tenets for controlling the behavior of network. They divided their study in two parts, one offering the innovative ways for finding the traffic anomalies, reaction to threats, flexibility in policy formation and deployment. Second part of the work provides security mechanism build up using SDN analytics which can be applied to the networks in real time. But they do not discuss the SDN security issues and its comparison with TNs. Dacier et al [8] discussed the current security challenges and showed how the traditional network architecture cannot fulfill the today's network demands. They discussed the various opportunities and challenges for security advancements in SDN. But they did not provide any way or model for SDN threat analysis and their resolution. B. Ahmad et al [9] discussed about Flow Table Entry Attack (FTEA), a kind of DoS attack, when Flow Entry Table gets full it drops the incoming packets or remove the prior flows. They assume that the attacker has access to SDN domain and consumes the controller resources by constantly engaging it to install attacker initiated bogus entries in the FET. However it exhibits only switches attack controller use case. Our work covers the four most important use cases for SDN threat analysis and their countermeasures.

III. Proposed Threat Model

Based on the SDN architecture we have derived a threat model which reflects how the various threats can attack the SDN components. SDN components are interlinked with each other if one component is compromised; it is a threat to another component and even for whole network. Our goal here is to identify the various attacks which can be performed by the attacker on a particular component of SDN. These components are SDN applications (Application Plane), controllers (Control Plane) and networking devices e.g. switches (Data Plane). In Fig. 3 we have shown the block diagram of SDN featuring its components. Based on this architecture we have derived four use cases to analyze the threats.

Threat Model:

There are many ways to exhibit SDN security issues and their resolutions [10][11]. Most of the authors discuss the same with layer based approach but we believe SDN architecture is different aspect from conventional network and we define a new taxonomy to generalize the SDN security issues. We consider a network scenario where there are n no. of controllers $C = \{c1,c2, \ldots\ldots.c_n\}$. Each controller ci □□ C can run at least one application from a set of applications $A^{ci} = \{a1.a2, \ldots\ldots a_n\}$. Each controller has limited resources which make them vulnerable to denial of service attacks. We have derived four use cases from SDN architecture. Each of the use case has its own importance and security goals. Fig. 4 shows the Threat model for security requirement of SDN. SDN architecture with associated use cases is shown in Fig. 3 and 4 respectively. A semi benign attack is a passive attack which may gather information about network or processes but will not deviate from protocol execution. A malevolent behavior is an active threat which may deviate from protocol rules in order to disrupt the system and attack the other components of the system [12][13]. These four use cases are described as under.

A. Use Case 1: Securing Controller from Applications in Application Plane

In this use case each application in the application plane can be benign, semi benign, or malevolent. These applications may be from different sources i.e. third party apps [14]. The controller proffers an abstraction to application plane so that application can read/edit network state which is generally a degree of network control. If an attacker impersonates application it can gain access to controller i.e. network control and can hamper the network operations [15]. The absence of trust and weak authentication between applications and controllers may lead to spoofing attacks [16][17]. Our goal here is to minimize the attacks on controllers through applications. List of such type of attacks and suggested solutions have been shown in Table I.

B. Use Case 2:Inter Controller Security

In SDN, control is logically centralized. It provides more than one controller for providing scalability and avoiding single point of failure [18]. As a result these controllers share the resources and communicate with each other. It is necessary to review the security of inter controller communication [19]. In this use case we assume one or more controller is semi benign or malevolent. A semi benign controller could be able to access the control data of other controllers, learn resource utilization information and target the integrity of the network. Moreover a malevolent controller can attack to semi benign controller and perform a DoS attack on another controller. Our goal is to protect controller from each other [20]. The possible attack scenario and solutions have been discussed in Table II.

C. Use Case 3: Securing Switches from Controller

In this use case it is assumed at least one controller is semi benign or malevolent. We assume that applications which are used through this controller can be semi benign or malevolent. A semi benign controller can target switches in the data plane. It can attack switch flow table with buffer overflow by sending bogus entry [21]. Our goal here is to eliminate the possibility of controller's ability to target the switch with bogus entry [22]. This case has been shown in Table III with threats and their solutions.

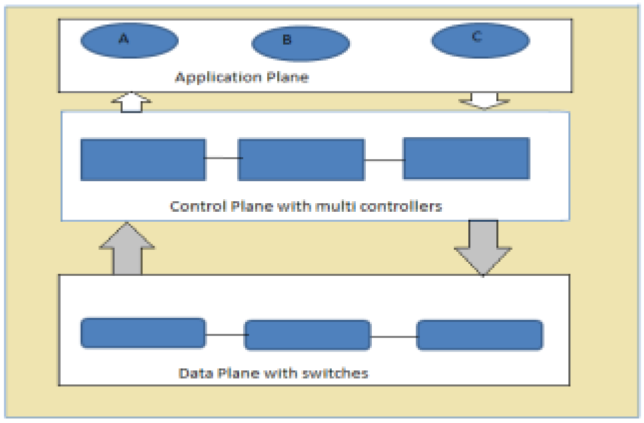
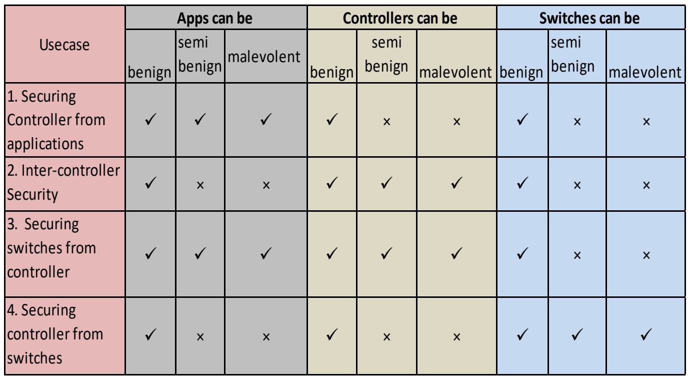


Fig. 3. SDN Architecture.



Usecase	Apps can be			Controllers can be			Switches can be		
	benign	semi benign	malevolent	benign	semi benign	malevolent	benign	semi benign	malevolent
1. Securing Controller from applications	✓	✓	✓	✓	×	×	✓	×	×
2. Inter-controller Security	✓	×	×	✓	✓	✓	✓	×	×
3. Securing switches from controller	✓	✓	✓	✓	✓	✓	✓	×	×
4. Securing controller from switches	✓	×	×	✓	×	×	✓	✓	✓

Fig. 4. Threat Model.

TABLE I. SECURING CONTROLLERS FROM APPLICATIONS (USE CASE 1)

Issues	**Possible Attack**	**Scenario**	**Possible Solution**
Application vulnerabilities	Remote code alteration and execution	An attacker can reprogram the application by using vulnerability and run the malicious code.	Periodic vulnerability scanning for applications
Use of Untrusted applications	Spoofing the messages between controller and applications	Absence of trust between controller and applications may lead to spoofing attack	Apps authentication and authorization must be implemented.
Inappropriate authorization	Unauthorized access to applications	If an application has weak authorization an attacker can gain unauthorized access to application and can attack the controller.	Use of AAA to protect the unauthorized access to application and controller

TABLE II. INTER-CONTROLLER SECURITY (USE CASE 2)

Threat	**Possible attack**	**Possible scenario**	**Possible Solution**
Untrusted Controller	Attack controller within cluster	Untrusted controller software can pose a serious threat to other controllers.	Use the trusted controllers provided by the trusted vendors.
Controllers configuration defects	Unauthorized access and network attacks	Providing unnecessary privileges to an app can result in controller hijacking.	To review the default configuration of controller and to do a proper controller hardening.
Embedded malware	Malware and spyware attacks	Spyware and ransomware gaining control to controller	Monitoring and scanning the network with trusted security applications.
Vulnerabilities within controller runtime	Controller runtime attacks	In case of third party network applications, vulnerabilities at run time can allow applications to modify its default configuration.	Controller's software should be updated periodically with new patches and updated versions.
Poorly separated inter-controller traffic	DOS attack, ARP spoofing	In a multi controller environment If traffic between controllers is poorly separated than it can allow a compromised controller to perform a man in middle or DoS attack.	Controllers in cluster should not be provided unnecessary permissions and should be monitored as per cluster rules

TABLE III. SECURING SWITCHES FROM CONTROLLER (USE CASE 3)

Threat	**Possible attack**	**Possible scenario**	**Possible Solution**
Controller switch communication channel.	Flooding attack on switch flow table.	A compromised controller can flood a switch by bogus entries by sending fake packets to switch.	The best way to secure the controller and switch communication is use of TLS.
Malicious applications	Attack on switches due to malicious application	An infected application may affect the controller and attack the switch through misconfiguration	Use of trusted and stable application for performing the network operations in SDN

D. Use Case 4: Securing Controller from Switches

In this case at least one switch is semi benign or malevolent and it tries to attack the controller [23]. An attacker can send fake message through this compromised switch to controller and tries to exhaust the controller's resources [24]. This condition is called as data leakage where attacker tries to discover the flow rules and forwarding policy information. If an attacker can gain access on packet processing timings and can determine the action related to specific type that are forwarded to controller, attacker can produce the phony flow messages causing to DoS attack [25][26]. Our goal here is to protect controller from switches. If a switch goes out malevolent or semi benign there should be a mechanism to find out the malicious switch in the network. One of the recent researches towards malicious switch detection has been presented in [27]. The authors presented a new algorithm to find a pernicious switch based on control path routing approach. This method chooses two node disjoint control path for every forwarding device in data plane so that a suspicious node can be find out on basis of simple Packet_In messages delivered to control paths. In [28] a novel technique for detecting the link flooding attack has been presented. Authors designed LFA defense system called LFADefender using SDN which contains features like programmability, complete view of network and flow traceability. In LFADefender, authors proposed a LFA target link selection approach and design a LFA congestion monitoring mechanism to effectively detect LFA.

IV. COMPARATIVE ANALYSIS OF SDN THREAT MODEL USE CASES WITH TRADITIONAL NETWORK

Based on the above use cases we have identified the attack scenario of various threats in SDN. Now we will compare the same with traditional network architecture to find out that; are these use cases available there in traditional networks? We will Fig. 5 out if these use cases are available in traditional network how we counter them. Then we will identify the SDN protection mechanisms [29][30] based on this. Traditional network architecture with four routers and two switches has been shown in figure. In Traditional Network (TN) the interface between two routers is called network to network interface (NNI) while the router interface with end user is called user network interface (UNI).

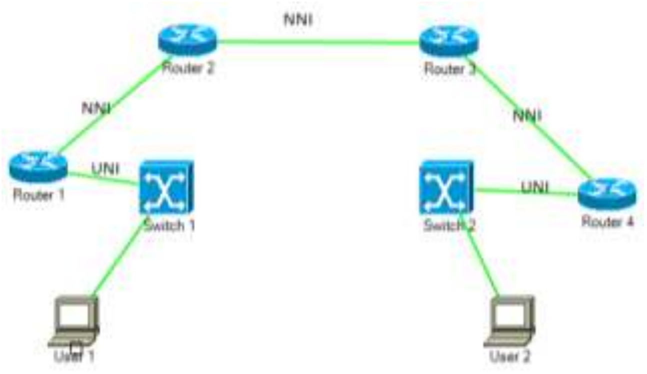


Fig. 5. Traditional Network Architecture.

The fundamental difference between SDN and TNs is control plane [31]-[32]. In TNs network controlling elements are inside the network devices e.g. routers and switches but in SDN it has been decoupled from the devices to a central controller. From the previous use cases in Section II, we can derive that controller security is most important and it can be attacked by applications, by switches in data plane and can even by other controller in multi controller environment [33]. In this section we will find how the network controlling elements are protected in tradition networks. What types of the attacks are faced in TNs and what are the protection mechanisms. We will try to analyze four use cases in TNs which we implemented in SDN in above section and it will give a clear picture of security problems and challenges in SDN with their possible solutions.

A. Use Case 1: Application Attacks Controller or Controlling Elements

If we talk about the application layer of SDN in terms of traditional network and its applications like mailing, FTP and HTTP etc. then we realize that these are the network functions for which lower layers in TCP/IP model have been designed. We cannot compare these functions with applications in application plane in SDN [34]. Applications in SDN are network controlling elements which have been decoupled from devices like routing, switching, security etc. and decoupled control functions works as applications and perform related network operations in coordination with controller. In TNs applications are the part of network devices and controlling part resides inside the devices. Hence application attacks controller use case 1 does not apply in TNs. However TNs do not provide any programmability to control the behavior of network dynamically like SDN which provide this functionality through applications in northbound API with an alternative to use third party network applications to customize the network as per demands [35].

B. Use Case 2: Controller Attacks other Controller

A controller in SDN performs the network control functions like routing switching etc. In TNs a routing function is performed by routers. A multi controller scenario in SDN can be compared with TNs having multiple routers. In network containing more than one router and various links from one router to another, a routing protocol is used to find the best path from source to destination. There are two types of protocols one distance vector routing protocols Routing Information Protocol (RIP) which find the best path to a remote network by judging distance. Second is link state routing protocols e.g. Open Shortest Path First (OSPF). These routing protocols maintain a routing table and contain the information about the neighbor subnets and links state. The router updates it routing table and advertise the routing information time to time and a best path is selected based on this information. However different routing protocols suffer from different attack methods but the common objective is to pollute routing tables. A routing table poisoning attack is performed to contaminate routing table and network topology information by advertising or infusing a bogus route through announcements. A malevolent router can publish a phony link state advertisement (LSA) with fake link cost to effect rest of the routers routing table calculations. This type of attacks are not difficult to dispatch but has limited influence as adjoining router will publish a right LSA with a new sequence entry which will remove the bogus LSA and it will not be used again for routing table route estimation. However a more powerful attack can also be performed to pollute the functioning routing table. To mitigate routing table poisoning, a routing protocol should only run on NNI and route advertisements from UNI need to be discarded. The origin of message should be checked for authentication to forestall a vindictive router to imitating another router. And routing updates need to be double checked before applying them for route estimation by routing tables. For example a link metric updated by one router need to be double checked with link metric updated by other router on the same link. Such type of defense techniques [36] can also be considered in SDN and the same has been discussed in Section II.

C. Use case 3: Controller Attacks Switches

In traditional network we can refer the L3 devices as control plane. Now we will try to find out, Can a router attack on the functionality of L2 devices? When a router sends a packet to another router, the receiving router performs three operations. First it eliminate the L2 header of packet, second check the routing table for next router in sequence and third bundles L2 header of packet for sending to next node. The next node in the routing table might be a local interface or it is an IP address. This routing process is continued till the next node is a local interface. Upon finding a local interface it will looks up for MAC address in ARP table of that interface. If it is unable to get the MAC the router will run the ARP protocol to get the MAC address associated to respective IP address. Because address resolution protocol (ARP) is used by router to find the MAC address associated with an IP address, an L3 router may suffer with ARP cache poisoning attack [37]. In this attack the attacker associates its own IP address with victim MAC address and receives the traffic intended to for victim node. So an attack can be placed from control devices i.e. from control plane on L2 traffic in traditional networks.

D. Use case 4: Switches Attack Controller

In SDN there is no by default communication when an open flow switch receive a new packet it sent it to the controller using Packet_In message, which includes source and destination address. If destination MAC address is not known by the controller then controller asks the switch to broadcast the packet through Packet_out message. The destination sends the response to the source port and this reply also noted by controller to fulfill the further requests from same source and destination. This process is called host tracking service and is equivalent to L2 MAC learning, in principle, with only difference that MAC learning has been separated from switch and included in controller. In SDN data plane switches communicate with controller for L2 learning process and can attack the same as discussed in Section II. But in TNs L2 learning is implemented inside the switches without any controller. So attack on L2 learning is equivalent to attack on controller from switches. A MAC table is learned from data plane switches including host packets so it is subjected to MAC attacks. An infected host can send a packet with fake MAC address to poison switch's MAC table. MAC spoofing and MAC flooding are two strategies of attacks which affect the L2 learning in traditional network [38]. Threats related to MAC address can be minimize by disallowing the unknown devices to enter the network. This can be done by a switch feature port security. Port security is a technique which allows only known MAC addresses (MAC binding) to be recognized by the network switches). But there is a limit to bind the number of MAC address associated with a switch port. But port security requires a lot of manual configurations which leads to possible overhead and misconfigurations [39].

V. Lessons Learned and Security Enhancements in SDN

We have compared four most important attack use cases in SDN and traditional network. We have seen how the control functions in TNs can be attacked in different use cases as compare to attacks on controller in SDN [40]. Table 4 shows the comparison of threat use cases in SDN and TNs. Now we will discuss the lessons learned on comparing these use cases in terms of threats and their defenses. We will explore how the security can be enhanced [41] in SDN based on our threat model. First we will elaborate each use case then a security application will be developed based on attacks from above use cases.

A. Use Case 1: Securing Controller from Applications

As we have discussed in previous section that network control functions are part of network devices in traditional network hence this case, does not apply on TNs. In SDN network control functions are in the form of applications and have been decoupled from network devices. These applications work for the data plane devices in coordination with controller [42]. As a matter of fact these applications communicate with controller to fulfill the network requirement and an unauthorized application can do a big damage to the controller and even reconfigure the network [43]. In order to counter an unauthorized application access controller and application should maintain a trusted connection and authenticate the identity of entities before exchanging control messages. Both authentication and authorization of applications is to be ensured before establishing a connection. This concern about the untrusted applications authentication and securing the controller has been discussed in [44]. Authors introduced a hierarchical arrangement of controllers. This hierarchical system can minimize the effect of pernicious application as code of the application would run at the middle hierarchy where there will be ample protection. Another work in this direction is FortNox [45]. FortNox is an extension to the open source controller NOX [29]. It is a security enforcement kernel which checks the flow rules for security policy violation in real time. Each openflow application is provided authorization through a role based authentication concept. Three flow rule producer roles are defined; OF Operator, OF Security, and OF Application. In case of any flow rule conflict detected by FortNox, a higher priority rule is accepted. The limitation of FortNox is application identification and priority enforcement. ROSEMARY [46] is the enhancement to controller resilience to malicious applications. It is a high performance network operating system which is robust and secure. It sandboxes the each running instance of application to provide security to control layer from any vulnerability. It also monitors and control the resources consumed by each application. In LegoSDN [47] authors explore about the effect of application failure on controllers reliability. Authors proposed a isolation layer between controller and applications to avoid the consequences of failure of controller due to application failure.

B. Use Case 2: Inter-controller Protection

In SDN to avoid the single point failure a multiple controllers has been suggested. There are two types of controller placement schemes; one is flat controller deployment and another is hierarchical controller deployment. In flat controller concept each controller is assigned a separate sub network. In this solution different operations may not be able to communicate equally with different domains. But in hierarchical mode the local controller is responsible for respective network, and global controller is responsible for local controller. The communication among different controllers is done via global controller. A variety of works has been done towards controller placement problem. In [48] authors proposed an algorithm to find the minimum number of controllers and maximum load on a controller. But this arrangement did not work for the request with variable time. In [49] author proposed an algorithm which divide the network in to different subnets. Every small network contains a controller based on the size of assigned network. It uses a clustering algorithm based on switch density, and divides the network accordingly. When the main link is broken it may use a backup link. But it may result in unnecessary delay. In [50] authors provide a multi controller solution with Byzantine fault tolerant mechanism. When one controller goes down, the other controller takes the charge of network and removes idle link of previous controller. However this solution is good for small network due to performance issues in relatively large networks.

C. Use Case 3: Protecting Switches from Controller

In traditional network, a control element router can attack the switch functionality through the ARP spoofing attack as discussed in Section III. But in SDN controlling element controller has more functionality and a malicious controller can do a lot of damage to the switches of data plane. A compromised controller can attack the switch flow table by generating unnecessary broadcast and overflow the switch flow table. So protecting the controller to become malicious is the main defense for data plane switches. In [26] authors proposed a solution for detecting the malevolent SDN device in the network. They implemented a backup controller and collect the state information and updates from primary controller and switches. They detect the malicious devices by recognizing the unexpected and inconsistent behavior of primary controller, backup controller and SDN switches.

TABLE IV. Comparison of Attacks use Cases

Use Cases	SDN	Traditional Network
Applications attack controller	A malicious application can attack controller.	Not applicable
Inter controller attack	A compromised controller can attack the other controller in a multi controller environment.	A router can attack the other router and can pollute its routing table by fake LSA.
Controller attacks switches	A malicious controller can attack the data plane switches and launch flood attack.	A controlling element router can attack the L2 switches by ARP cache poisoning attack.
Switches in data plane attacks on the controller	A malevolent switch can flood the controller by fake flows may results in DoS attack.	In L2 network ,the control function MAC learning, can be targeted by MAC spoofing and MAC Flooding attacks.

D. Use Case 4: Protecting Controller from Switches

The main protocol which provides the interface for communication between data plane switches and controller is open flow. In respect to southbound interface communication, the open flow switch specifications discuss necessity of TLS with mutual authentication between controller and switches [51]. In [52] the lack of TLS adoption is real world deployments has been discussed. This is very important consideration when switches, controller and application environment is deployed in trust domains. However it is to be noted here that there is definite weakness introduced by separating the control and data plane in SDN [53]. Various solutions to avoid the DoS attack have been proposed. AVANT-GUARD [48] provides the protection to the controller from switches by limiting the number of flow requests sent to the controller by using a connection migration tool. This migration tool removes failed TCP sessions at the data plane prior to any notification to controller. This prevents the occurrences of DoS attack by sending only those flow requests to the controller which completes the TCP handshake.

E. Security Enhancements

By the comparisons and discussion in the last two sections it can be stated that there is a need to develop a security mechanism to counter the security issues of SDN. As discussed that the controlling functions in the SDN are performed by the applications in application plane. For implementing the security functions there is a need to design the security application in SDN. In this section we develop a security application as a part of SDN software. In traditional network if we want to implement security functions then we need to use a hardware device e.g. firewall for the same. But this is advancement in SDN that network controlling functions like security, routing, and monitoring etc., are in the form of applications. For Design and implementation, we will use mininet as network emulator and Ryu as a controller. First we will focus basic steps and algorithm for designing an application as per controller and data plane communication.

Python language is used to develop the network applications based on Ryu controller. Ryu is a components based controller which has various modules for application design and control. In ryu controller setup at home/ubuntu/ryu it has various folders; app, base and ofproto. App folder can contain various applications like firewall, router and load balancer. Base folder contains App_manager which helps to run the different applications and prepares framework and datapath for running the application. Ofproto deals with openflow version related queries and matching capabilities. For designing a SDN application we need to collect and understand the initial requirements and booting process of SDN network framework.

- In first step switch boots up and contact the controller for openflow version related queries and check its capabilities.
- The controller installs Packet In function and table miss function and prepares itself for queries from switch.
- When receiving Packet In, Controller learns the source MAC and mention the MAC and port information in flow table. It checks for destination MAC address if it is available in flow tables, it uses Packet Out function on the port and installs the flow and stores the same for future uses.
- If destination MAC address is not available in flow table i.e. a table miss then controller uses packet out function to broadcast the packet to all ports.

By using the ryu controller framework we can design and deploy customized security applications. With programmability approach in SDN we can have our own security application in ryu app folder and program it as per network demands and configure it through standard API. Traditional security solutions, the vendor specific e.g. fortigate and Cisco, they have their own proprietary code and configuration methods which are fixed and cannot be customized as per demands. Fig. 6 shows how the security app can work in coordination with controller. When Host A wants to communicate to Host B it sends a packet to switch. Switch check for a matching entry in its flow table but when a matching entry is not found in flow table then packet is forwarded to controller. Controller sends the packet to security application for policy check. First it parses the packet and check if it matches to policy specified in firewall. As firewall has a policy to block traffic from A to B (A-->B: Block). The application enforces a rule through controller to drop the packet and controller install a flow rule in switch flow table to drop all the incoming traffic from Host A to Host B. This is how we can block and allow flow in openflow through a security application. It means through this app a switch can work like a firewall i.e. technology allows us to decide the functions of a switch. As a result additional security devices are not required in SDN as security services can be enabled within the devices. In traditional network another problem is placement of firewall for optimized coverage of security services. But it has been nullified as any device in the network can be turned into a security device.

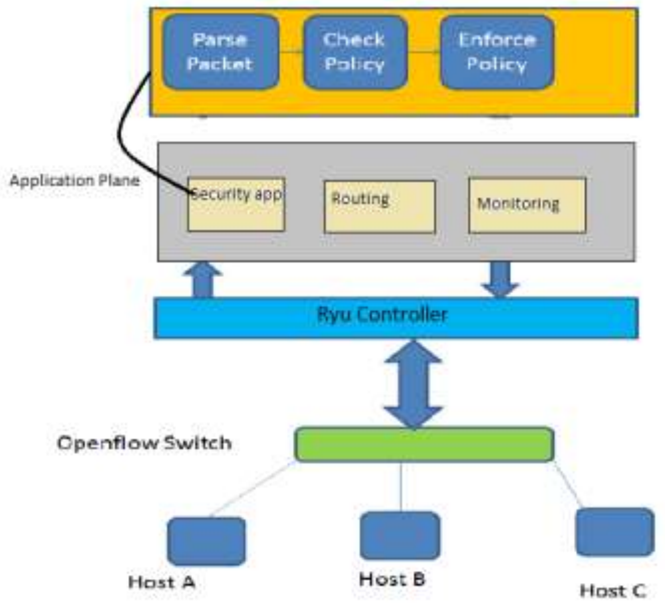


Fig. 6. Implementing Security Application in SDN.

VI. Future Research Directions and Proposed Security Architecture

Based on use cases presented in the paper and comparative analysis with traditional network it may be admitted that SDN has introduced some new security issues and challenges that are not available in TNs. But programmable networks include a level of adaptability and dynamic control that has enhanced the network management and flexibility at scale. SDN has emerged as a technology which may be taken as a replacement of vendor based proprietary CLI networking devices where there is no scope of programmability and automation due to tightly coupled software and hardware concept of networking devices. So it is to be noted that SDN is going to be stay here and some more potential research proposals are required to address the challenges of SDN security issues. Implementation of mandatory TLS functionality a controller and deplane communication channel can solve a lot of problems. Research proposals like AVANT-GUARD [48] has provided a good work by limiting the rate of number of requests sent to controller which improve the controller performance. Implementing some intelligence function to data plane switches may be considered to minimize the controller load. Such type of proposals is under discussion with research community in the form of stateful data planes [54].

Another important area which needs the momentum in research proposals is application-controller interface [55]. Without the presence of a standard open north bound API, it is not possible to design and deploy SDN in enterprise network [56]. The security enhancements explored in Section IV are of no means if application-control interface is vulnerable. This can also be evaluated from Table I of our threat model analysis that the switches in data plane can be attacked if either the applications or controller are malicious. In contrast we have discussed the various innovative proposals which analyses the protection requirements of north bound API. However this use case (securing controller from application) exhibits a lot of vulnerability to various attacks as discussed in Section II. As a results further research in this area are necessary and need to be encouraged for finding a better northbound API. However use of RESTful API [6] is also a good work and this may be extended further. A multi controller solution for addressing the scalability issue of controller has been provisioned in openflow 1.3. Various controllers need to communicate with controller in other domain for performing various operations to fulfill the network requirements [57]. A secure and real time communication of controllers is an open research problem. However a number of solutions have been discussed in section Vth in view of further research directions in this area [58]. A framework for network security application development has been presented based on ryu controller and mininet. This work can also be further explored by adding more security functions if we have a new idea and algorithm as per demands.

A. Proposed Security Architecture

By threat model analysis it can be pointed out that SDN security problem is not a problem of single SDN component, it is scattered in all components of architecture and these components are inter linked with each other and form a system. So there is a need to design a security solution as per system perspective rather than security for individual component. Based on our analysis security architecture for SDN has been proposed in Fig. 7.

Control-application interface is protected with AAA security at application plane. We believe each application should be developed as a module of controller so that it can easily follow the security standard of northbound interface designed to secure the communication. Even third party applications should follow and support the security policy standard at application-controller interface. A multi controller solution with hierarchical control is provided to avoid the single point failure of controller and resource sharing. A backup controller has also been proposed for global controller fault tolerance. At southbound API the communication of controller and data plane switches should be secured with mandatory TLS security function. For minimizing the load of controller some state level intelligence is suggested in data plane switches i.e. stateful data plane [53]. However the management of states and packet level forwarding decisions are taken from controller. Finally it is to be stated here controller security is the prime tenet to secure overall SDN platform and this is ultimately depends of secure applications environment at northbound API. Development of a standard northbound API is still an open research problem. Contribution haven been made in the form of RESTful API but a more research proposals are required in this direction to form a more secure SDN network.

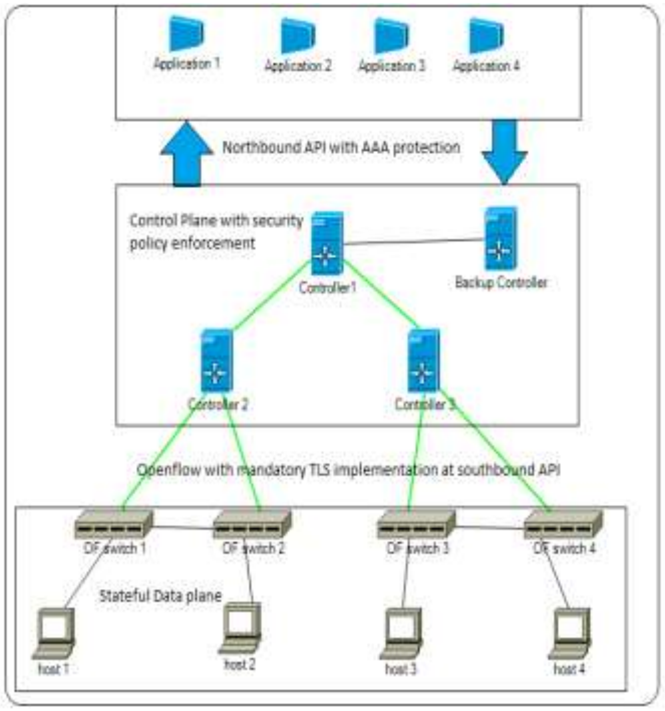


Fig. 7. A Proposed Security Model for SDN.

VII. Conclusion

To identify the SDN security issues we developed four use cases and discussed the several attack parameters with their counter measures in tabular format. After identifying the security issues we applied the same use cases in traditional network for a comparative study of risk and security technology in both the networks. After comparative study it can be concluded that SDN has introduced new attack surfaces which is not available in traditional networks. In contrast SDN provides more flexibility, automation and control over the network, traditional networks disappoint there. However security solutions to address the SDN security issues have been presented which includes protection from malicious application, inter-controller protection, protection of data plane and protecting controller from DoS attacks by data plane switches. Based on analysis a framework for development of SDN security application has been presented with ryu controller and mininet network emulator. Insights for security enhancement have been provided by presenting a proposed security model based on recent research and threat model analysis. Moreover research in SDN security is still in beginning stage and there is lot more to do with. By designing novel security techniques and extending the previous research work for solving known problems, we can find the better SDN networks which will be much more secure than traditional networks.

References

[1] M. Casado et al., “SANE: A protection architecture for enterprise networks,” in Proc. USENIX Security Symp., 2006, p. 10.

[2] M. Casado et al., “Ethane: Taking control of the enterprise,” in ACM SIGCOMM Comput. Commun. Rev., vol. 37, no. 4, pp. 1–12, Oct. 2007.

[3] N. McKeown et al., “OpenFlow: Enabling innovation in campus networks,” ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Apr. 2008.

[4] R. R. Yadav, E. T. G. Sousa and G. R. A. Callou, “Performance Comparison between Virtual Machines and Docker Containers”, IEEE Latin America Transactions, VOL. 16, NO. 8, AUG. 2018, pp. 2282-2288.

[5] S. Jain et al., “B4: Experience with a globally-deployed software defined WAN,” in Proc. ACM SIGCOMM Conf., 2013, pp. 3–14.

[6] Li Li, Wu Chou, Wei Zhou and Min Luo, “ Design Patterns and Extensibility of REST API for Networking Applications”, IEEE, TNSM, 2015, 00814.

[7] S. Taha Ali et. al “A Survey of Securing Networks using SDN”, IEEE transactions on reliability, Vol 64, No. 3, 2015.

[8] Marc C. Dacier et al, “Security Challenges and Opportunities of Software Defined Networking”, in *IEEE Computer and Reliabilities Societies*, 2017, pp.96-100.

[9] B. Ahmad et al. “Fingerprinting SDN policy parameters : An Empirical Study”, IEEE Access, Volume 8, 2020.

[10] D. Li, X. Hong, and J. Bowman, “Evaluation of security vulnerabilities by using ProtoGENI as a launchpad,” in Proc. IEEEGLOBECOM, 2011, pp. 1–6.

[11] S. Shin and G. Gu, “Attacking software-defined networks: The first feasibility study,” in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 165–166.

[12] L. Schehlmann, S. Abt, and H. Baier, “Blessing or curse? Revisiting security aspects of software-defined networking,” in Proc. 10th Int. CNSM, 2014, pp. 382–387.

[13] S. Sezer et al., “Are we ready for SDN? Implementation challenges for software-defined networks,” IEEE Commun. Mag., vol. 51, no. 7, pp. 36–43, Jul. 2013.

[14] W. Han, H. Hu, and G.-J. Ahn, “LPM: Layered policy management for software-defined networks,” Data and Applications Security and Privacy XXVIII. Berlin, Germany: Springer-Verlag, 2014, pp. 356–363.

[15] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, “Towards a secure controller platform for OpenFlow applications,” in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 171–172.

[16] S. Scott-Hayward, C. Kane, and S. Sezer, “OperationCheckpoint: SDN application control,” in Proc. 22nd IEEE ICNP, 2014, pp. 618–623.

[17] P. Porras, S. Cheung, M. Fong, K. Skinner, and V. Yegneswaran, Securing the software-defined network control layer,” in Proc. NDSS, San Diego, CA, USA, Feb. 2015, pp. 1–15.

[18] P. Berde et al., “ONOS: Towards an open, distributed SDN OS,” in Proc.3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp. 1–6.

[19] M. M. O. Othman and K. Okamura, “Securing distributed control of software defined networks,” Int. J. Comput. Sci. Netw. Security, vol. 13, no. 9, pp. 5–14, Sep. 2013.

[20] F. Botelho, A. Bessani, F. M. Ramos, and P. Ferreira, “On the design of practical fault-tolerant SDN controllers,” in Proc. 3rd EWSDN, 2014, pp. 73–78.

[21] H. Mai et al., “Debugging the data plane with anteater,” ACM SIGCOMM Comput. Commun. Rev., vol. 41, no. 4, pp. 290–301, Aug. 2011.

[22] Ahmad, B. et al., “Fingerprinting SDN policy parameters : An Empirical Study”, IEEE Access, Volume 8, 2020.

[23] C. Jeong, T. Ha, J. Narantuya, H. Lim, and J. Kim, “Scalable network intrusion detection on virtual SDN environment,” in Proc. IEEE 3rd Int. Conf. CloudNet, 2014, pp. 264–265.

[24] S. A. Mehdi, J. Khalid, and S. A. Khayam, “Revisiting traffic anomaly detection using software defined networking,” in Recent Advances in Intrusion Detection. Berlin, Germany: Springer-Verlag, 2011, pp. 161–180.

[25] R. Braga, E. Mota, and A. Passito, “Lightweight DDoS flooding attack detection using NOX/OpenFlow,” in Proc. IEEE 35th Conf. LCN, 2010, pp. 408–415.

[26] J. Suh et al., “Implementation of content-oriented networking architecture (CONA): A focus on DDoS countermeasure,” in Proc. European NetFPGA Developers Workshop, Cambridge, U.K., 2010, pp. 1–6.

[27] Purnima Murali Mohan et. al., “Towards resilient in-band control path routing with malicious switch detection in SDN”, IEEE COMSNETS, 2018, PP.9-16.

[28] Haifeng Zhou et. al., ”SDN-RDCD: A Real-Time and Reliable Method for Detecting Compromised SDN Devices”, IEEE/ACM transactions on networking, vol. 26, no. 5, october 2018 pp. 2048-2061

[29] D. Kreutz, F. Ramos, and P. Verissimo, “Towards secure and dependable software-defined networks,” in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 55–60.

[30] S. Shin et al., “FRESCO: Modular composable security services for software-defined networks,” in Proc. Netw. Distrib. Security Symp., San Diego, CA, USA, 2013, pp. 1–16.

[31] N. Gude et al., “NOX: Towards an operating system for networks,” ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 3, pp. 105–110, Jul. 2008.

[32] D. Erickson, “The beacon OpenFlow controller,” in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 13–18.

[33] A. Guha, M. Reitblatt, and N. Foster, “Machine-verified network controllers,” ACM SIGPLAN Notices, vol. 48, no. 6, pp. 483–494, Jun. 2013.

[34] S. H. Yeganeh and Y. Ganjali, “Kandoo: A framework for efficient and scalable offloading of control applications,” in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 19–24.

[35] N. Foster et al., “Frenetic: A network programming language,” ACM SIGPLAN Notices, vol. 46, no. 9, pp. 279–291, Sep. 2011.

[36] T. Koponen et al., “Onix: A distributed control platform for large-scale production networks,” in Proc. OSDI, 2010, vol. 10, pp. 1–6.

[37] Seung Yeob Nam, Dongwon Kim and Jeongeun Kim. "Enhanced ARP: Preventing ARP Poisoning-Based Man-in-the-Middle Attacks" IEEE Communications Letters, Vol. 14, No. 2, February 2010, pp. 187-189.

[38] Songyi Liu, "MAC Spoofing Attack Detection Based on Physical Layer Characteristics in Wireless Networks" IEEE, ICCEM, 2015.

[39] Timo Kiravuo, Mikko S¨arel¨a, and Jukka Manner, "A Survey of Ethernet LAN Security" IEEE Communications Surveys & Tutorials, Vol. 15, No. 3, 2013,pp. 1477-1491.

[40] A. Zaalouk, R. Khondoker, R. Marx, and K. Bayarou, "OrchSec: An orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions," in Proc. IEEE NOMS, 2014, pp. 1–9.

[41] Pradeep Kumar Sharma and S.S Tyagi "Improving Security through Software Defined Networking (SDN): An SDN based Model", IJRTE, vol. 8, issue 4, 2019, pp. 295-300.

[42] Marcelo Ruaro , Luciano Lores Caimi , and Fernando Gehm Moraes, "SDN-Based Secure Application Admission and Execution for Many-Cores", IEEE Access, volume 8, 2020, pp. 177296- 177306.

[43] D. Kreutz et al., "Software-defined networking: A comprehensive survey," arXiv preprint arXiv:1406.0440, 2014.

[44] D. Yu, A. W. Moore, C. Hall, and R. Anderson, "Authentication for resilience: The case of SDN," in ser. Security Protocols XXI. Berlin, Germany: Springer-Verlag, 2013, pp. 39–44.

[45] P. Porras et al., "A security enforcement kernel for OpenFlow networks," in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 121–126.

[46] S. Shin et al., "Rosemary: A robust, secure, and high-performance network operating system," in Proc. ACM SIGSAC Conf. Comput. Commun. Security, 2014, pp. 78–89.

[47] B. Chandrasekaran and T. Benson, "Tolerating SDN application failures with LegoSDN," in Proc. 13th ACM Workshop Hot Topics Netw., 2014, p. 22.

[48] G. Yao, J. Bi, Y. Li, et al., "On the Capacitated Controller Placement Problem in Software Defined Networks", IEEE Communications Letters,vol.18, no.8, 2014, pp. 1339-1342.

[49] J. Liao, H. Sun, J. Wang, et al., "Density cluster based approach for controller placement problem in large-scale software defined networkings", Computer Networks, vol.112, 2017, pp. 24-35.

[50] H. Li, P. Li, S. Guo, et al., "Byzantine-resilient secure software-defined networks with multiple controllers", Proc. IEEE International Conferenceon Communications, 2014, pp. 695-700.

[51] OpenFlow Switch Specification Version 1.4, Open Network.

[52] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 151–152.

[53] Josy Elsa Varghese and Balachandra uniyal, "An efficient IDS framework for DDOS attacks in SDN environment", IEEE Access, 2021, pp. 69680-69699.

[54] Tooska Dargahi et. al., "A Survey on the Security of Stateful SDN Data Planes" IEEE Communications Surveys & Tutorials, Vol. 19, No. 3, 2017, PP. 1701-1724.

[55] A. A. Z. SOARES et. al., "3AS: Authentication, authorization, and accountability for sdn-based smart grids ", IEEE Access, volume 9, 2021, pp. 88621-88640

[56] Kevin Barros Costa et al., "Enhancing Orchestration and Infrastructure Programmability in SDN with NOTORIETY", IEEE Access, Volume 8, 2020, pp. 195487-195502.

[57] Basem Almadani , Abdurrahman Beg and Ashraf Mahmoud, "DSF: A Distributed SDN Control Plane Framework for the East/West Interface" IEEE Access, Volume 9, 2021, pp. 26735-26754.

[58] Ahmed Sallam , Ahmed Refaey,and Abdallah Shami, "On the Security of SDN: A Completed Secure and Scalable Framework using the Software-Defined Perimeter", IEEE Access, volume 7, 2019. pp. 146577-146587.