FUDMA Journal of Sciences (FJS)
ISSN: 2616-1370
Vol. 2 No. 2, June, 2018, pp. 28 - 36

A SECURITY ARCHITECTURE FOR SOFTWARE DEFINED NETWORK (SDN)

***1Okunade Oluwasogo Adekunle and 2Osunade Oluwaseyi**

Department of Computer Science, Faculty of Sciences, National Open University of Nigeria,
91, Cadastral Zone, Nnamdi Azikiwe Expressway, Jabi, Abuja, Nigeria.
Department of Computer Science, Faculty of Science, University of Ibadan, Ibadan, Oyo State, Nigeria

ABSTRACT
Software defined network is emerging network architecture with promising future in network field. It is dynamic, manageable, cost effective, and adaptable networking where control and data plane are decoupled, and control plane is centrally located to control application and data planes. OpenFlow is an example of Software Defined Network (SDN) Southbound, which provides an open standard based interface between the SDN controller and data planes to control how data packets are forwarded through the network. As a result of rapid changes in networking, SDN program-ability and control logic centralization capabilities introduces new fault and easily attack planes, that open doors for threats that where not exist or harder to exploit. The paper present SDN architecture with security control level, this provide secured SDN paradigm with machine learning white/black list, where users application can be easily tested and group accordingly (malicious attack or legitimate packet).

Keywords-*Software Defined Network (SDN), OpenFlow, Flow Table, Security Control, White/Black List*

INTRODUCTION

Despite the fact that Internet has led to the creation of digital globalization; traditional Internet Protocols (IP) networks are complex and very hard to manage especially in the area of network configuration, according to the predefined policies and to reconfigure it to respond to faults, loads and changes. The basic concept of software defined network (SDN) is to separates the network control (brains) from forwarding (muscle) planes to make it easier to optimize (Haripriya and Sangeethalakshmi, 2015). Most common protocol used in SDN network to facilitate the communication between Controller and switches/routers is called OpenFlow. Southbound Application Programme Interface (API) is an interface between the Control and Data planes, this is where the protocols can be find. There are different types of protocols but the most commonly used among them is OpenFlow, it is an open standard communication protocol that enables the control plane to interact with the forwarding plane. People often point to OpenFlow as being synonymous with SDN, but it is only a single element in the overall SDN architecture. Figure 1, shows a traditional network of five devices with each comprising of a control plane that provides information used to build a forwarding table, application and forwarding table used to determine received frames or packets destination.

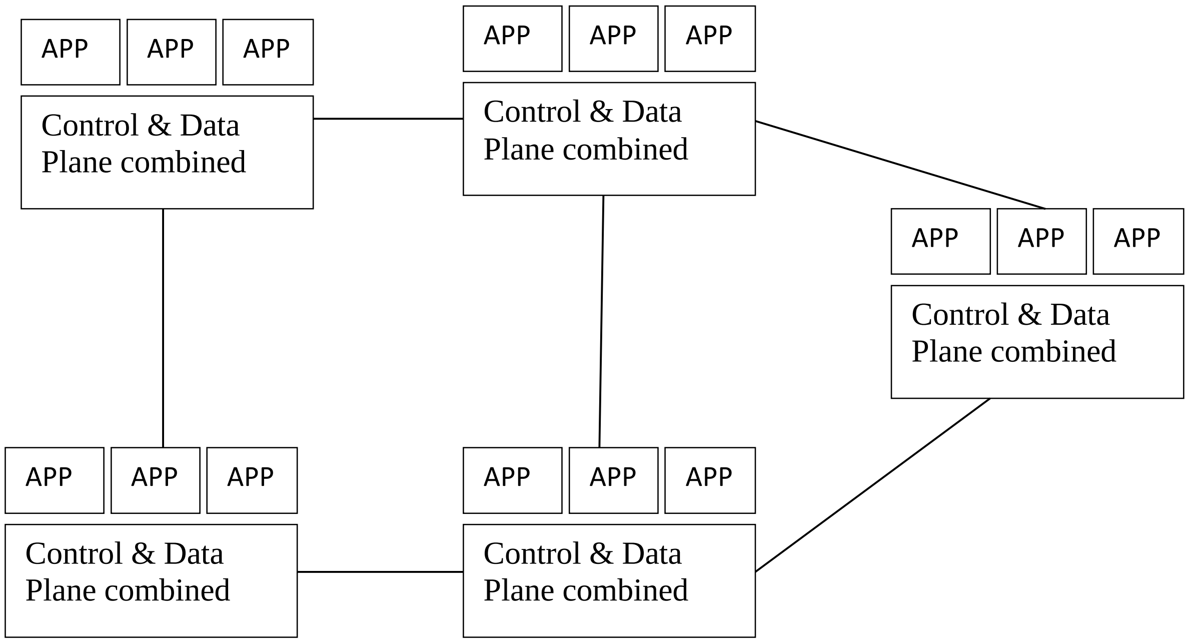

APP
APP
APP
Control & Data Plane combined
APP
APP
APP
Control & Data Plane combined
APP
APP
APP
Control & Data Plane combined
APP
APP
APP
Control & Data Plane combined
APP
APP
APP
Control & Data Plane combined

Fig. 1: Application planes separated from combined Control and Data planes of Traditional Network

In traditional networks, routers and other network devices encompass both data and control function. Making it difficult to adjust the network infrastructure and operation rather than the predefined policies regardless of faults, loads and changes that may later occurs (Furqan, Iyad and Ahmed, 2013). The control plane is an element of a router or switch that determines how each device within a network interacts with its neighbours. Examples of control plane protocols are; routing protocols, such as Open Shortest Path First (OSPF), Border Gateway Protocol (BGP), and Spanning Tree Protocol (STP). These protocols determine the optimal port or interface to forward packets (that is, the data plane). While the control plane protocols scale correctly, and provide a high level of network resiliency. They pose limitations for example, routing protocols may only able to determine the best path through a network based on static metrics such as interface bandwidth or hop count. Likewise, control plane protocols do not typically have any visibility into the applications running over the network, or how the network may be affecting application performance. Data plane functionality includes features such as' quality of service (QoS), encryption, Network Address Translation (NAT) and access control lists (ACLs). These features directly affect how a packet is forwarded, including being dropped. However, many of these features are static in nature and determined by the fixed configuration of the network device. There is typically no mechanism to modify the configuration of these features based on the dynamic conditions of the network or its applications. Finally, configuration of these features is typically done on a device-by-device basis (Mitchiner and Prasad, 2014), greatly limiting the scalability of applying the required functionality. While SDN abstracts this concept and places the control plane functions on SDN controller, where this controller can be a server running SDN software see Figure 3 where business requirements changes.

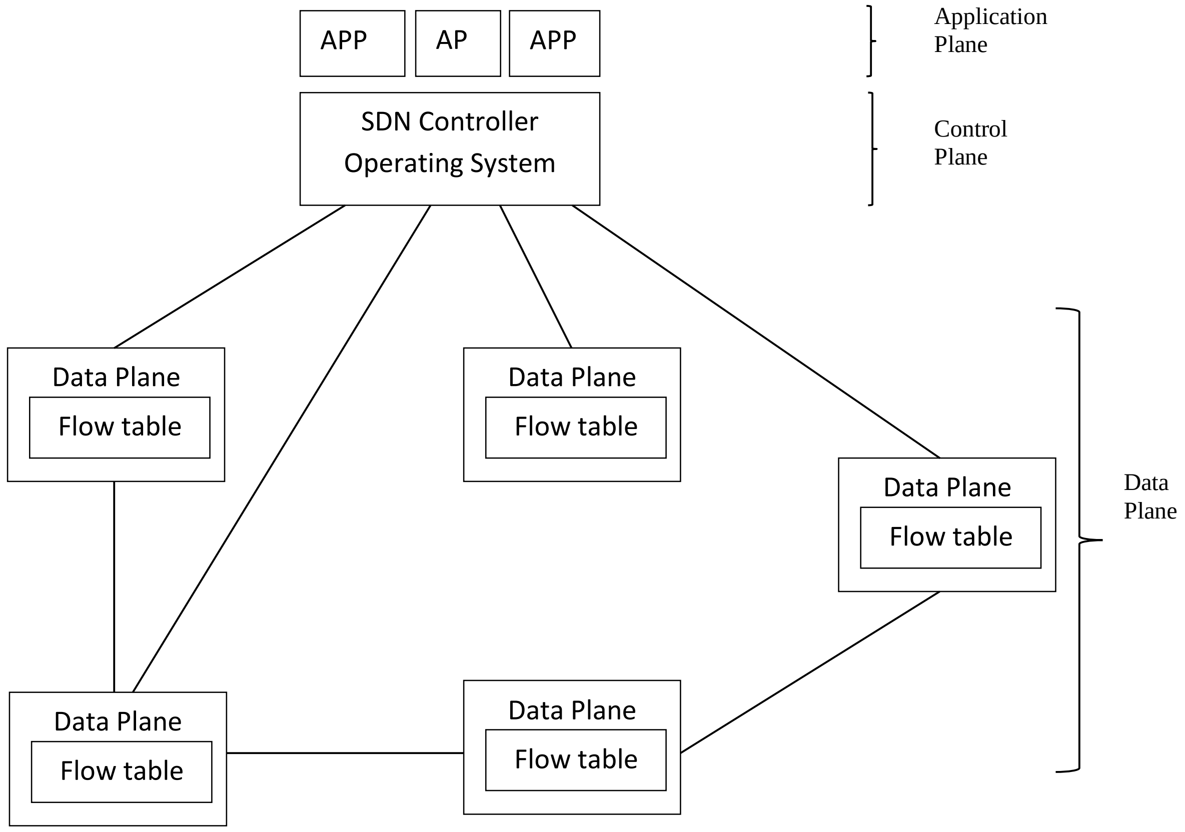


Fig. 2: Software Defined Network (SDN) with decoupled Control Plane from Data Plane

By using an API, your controller can implement network commands to multiple devices without the need to learn the command line syntax of multiple vendor products. These are few of the benefits seen with SDN. The control plane is responsible for configuration of the node and programming the paths that will be used for data flows. Once these paths have been determined, they are pushed down to the data plane. Data forwarding at the hardware level is based on this control information. Once the flow management (forwarding policy) has been defined, the only way to make an adjustment to the policy is via changes to the configuration of the devices. The change in the location and intensity of flows over time requires a flexible approach for successful network resource management. The numbers of handheld devices like smartphones, tablets, and notebooks have greatly increased the pressure on enterprise resources. Network resources changes rapidly and management of Quality of Service (QoS) security become challenging (Muhammad, Shyamala, Ali and Bill, 2014). In a security and dependability perspective, one of the key ingredients to guarantee a highly robust system is fault and intrusion tolerance (Diego, Fernando, Ramos and Paulo, (2013). According to Mark, Marco, Arjun and Nate (2013) Networks are expected to operate without disruption, even in the presence of device or link failures. However, Network programmability and control logic centralization capabilities introduces new fault and attack planes, which open the doors for new threats that did not exist before or were harder to exploit (Diego, et al., 2013). OpenFlow (OF) paradigm embraces third party development efforts, and therefore suffers from potential trust issue on OF applications (apps). The abuse of such trust could lead to various types of attacks impacting the entire network (Xitao, Yan and Chengchen, 2013). This can be seen as attractive honeypots for malicious users and major concern for less prepared network operators.

The ability to control the network by means of software (always subject to bugs and a score of other vulnerabilities) and centralization of the network intelligence in the controller(s) can make anyone with unlawful access to the servers (impersonation) potentially control the entire network unlawfully. The question now is; how can the Software-Defined Network be protected from malicious attack? Since potential security vulnerabilities exist across the SDN platform. At the controller-application level, questions have been raised around authentication and authorization mechanisms to enable multiple organizations to access network resources while providing the appropriate protection of these resources. However, with multiple controllers communicating or processing communication with a single, centralized controller, authorization and access control becomes more complex, potential for unauthorized access increases and could lead to manipulation of the node configuration and/or traffic through the node for malicious intent (Seungwon, Vinod, Phillip and Guofei, 2013). The remainder of this paper is organized as follows : literature review that reveals the status of other authors in the field, methodology that stated the approach applied to resolve the challenges, result gave the outcome of the research, discussion gave incite on the result gotten from the research and finally logical conclusion is made.

LITERATURE REVIEW

Software-Defined Network (SDN) creates an environment where all switches and routers take their traffic forwarding clues from a centralized management controller that communicates with network elements which can simplify manage and improve visibility into the network. SDN has the following three layers/plane;

1. Application Plane/Layer: Control layer implement logic for flow control

2. Control Plane/Layer: This runs applications to control network flows

3. Data Plane/Infrastructure Layer: this is a Data plane consists of the Network switch or router

The application layer contains network applications that introduces new network features, such as security and manage-ability, forwarding schemes or assist the control layer in the network configuration (Wolfgang and Michael, 2014). The application layer can receive an abstracted and global view of the network from the controllers and use that information to provide appropriate guidance to the control layer. The interface between the application layer and the control layer is referred to as the northbound interface. This is the interface through which the SDN Application layer communicates with the Control Layer to expose the program-ability of the network (Wolfgang and Michael, 2014). SDN controller manages the forwarding state of the switches in the SDN, this management is done through a vendor neutral API that allows the controller to address a wide variety of operator requirements without changing any of the lower level aspects of the network, including topology. With the decoupling of the control and data planes, SDN enables applications to deal with a single abstracted network device without concern for the details of how the device operates. Network applications see a single API to the controller. Thus it is possible to quickly create and deploy new applications to orchestrate network traffic flow to meet specific enterprise requirement for performance or security using API. Examples of north bounds interface are FML, Procera, Frenetic, RESTful and so on.

The OpenFlow protocol provides an interface that allows control software to program switches in the network, this is called southbound (Raj & Subharthi, 2013 and Wolfgang & Michael, 2014). Southbound is a protocol of OpenFlow which separates the control plane from the data plane to enable centralized and fine grained control of network flows. Examples of Southbound are OpenFlow, ForCES, PCEPNetConf, IRS and so on. OpenFlow is an example of Software Defined Networking (SDN), which provides an open, standards based interface to control how data packets are forwarded through the network, Controller communicates with a physical or virtual switch data plane through protocol that conveys the instructions to the data plane on how to forward data.

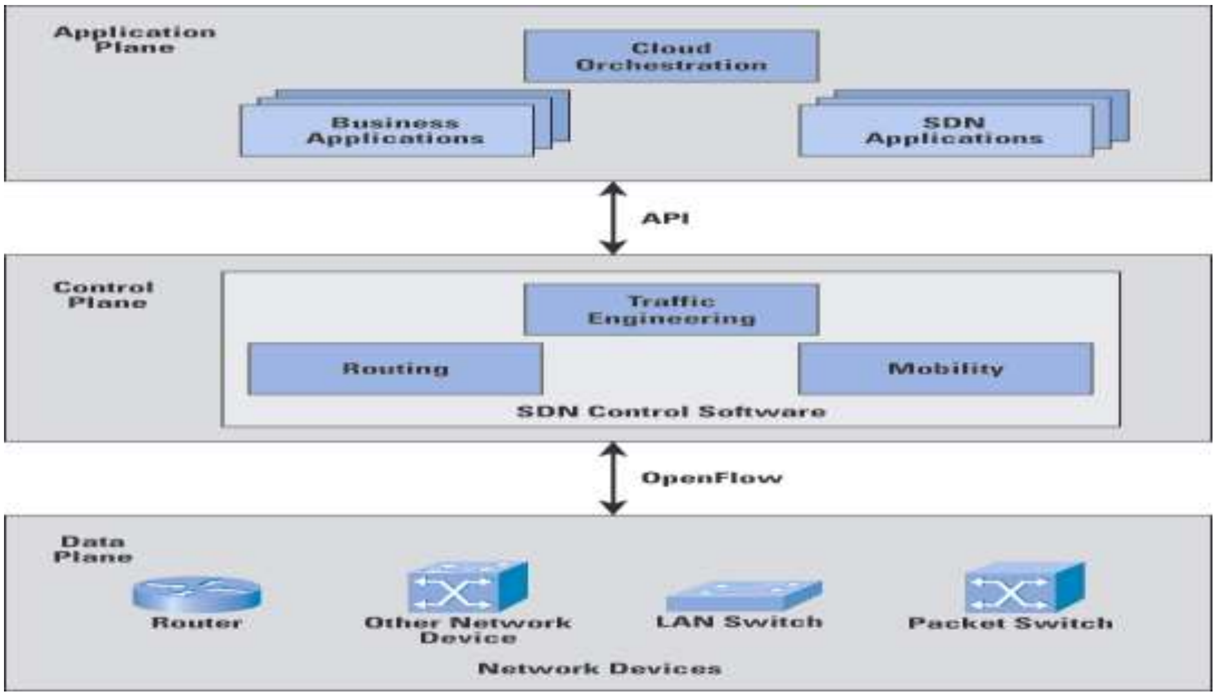


Fig. 3: Software Defined Network Architecture (Source: William, 2013).

This is a Software-Defined Network (SDN) package that enables networks to be software controlled, and used to dynamically change the network configuration, It is the most common example of southbound interface, which is standardized by the Open Networking Foundation (ONF). OpenFlow is a protocol that describes the interaction of one or more control servers with OpenFlow compliant switches. An OpenFlow controller installs flow table entries in switches, so that these switches can forward traffic according to entries. OpenFlow switches depend on configuration by controllers (Wolfgang and Michael, 2014). OpenFlow allows network switches to be configured using programmable interfaces, monitored/inspect network traffic and routing of packets (Yutaka, Hung-Hsuan and Kyoji, 2013). OpenFlow protocol specifies the interactions between the control plane running in the controller and the infrastructure; it is a foundational element for building SDN solutions. OpenFlow framework is an embodiment of the SDN concept, framework for the implementations of Software Defined Network (SDN) paradigm that enable communication between the controller and the switches uses a standardized OpenFlow protocol. In an OpenFlow environment, flow tables are used by devices rather than routing or MAC address table. Switches implement policy using efficient packet processing hardware: this is a secure channel that connects the switch to a remote control process (called the controller), allowing commands and packets to be sent between a controller and the switch using the OpenFlow Protocol Eddie et al, (2000) in McKeown et al, (2008). An OpenFlow network consists of a distributed collection of switches managed by a program running on a logically centralized controller, each switch has a flow table that stores a list of rules for processing packets and, each rule consists of a pattern (matching on packet header fields) and actions (such as forwarding, dropping, flooding, or modifying the packets, or sending them to the controller). OpenFlow Protocol provides an open and standard way for a controller to communicate with a switch (McKeown, et. al., 2008).

Controller machine manages a collection of programmable switches, defines the forwarding policy for the network and configures the switches through an open and standard (south bound) interface. A controller associates packets with their senders by managing all the bindings between names and addresses, it essentially takes over DNS, DHCP and authenticates all users when they join and keeping track of which switch port (or access point) they are connected to (McKeown, et. al., 2008).The controller derive the desired forwarding data in software, send OpenFlow messages to update the forwarding table in the device and the messages can add, update or delete entries in the forwarding table. Controller drives a level of network convergence; consider changing the entire configuration on your network to support new network path every 10 minutes. SDN Controller defines the data flows that occur in the SDN data plane: each flow through the network must first get permission from the controller, which verifies that the communication is permissible by the network policy. The controller is dynamical enough to make changes to network elements based on feedback or code without human intervention (Bruce, Rossi, 2016). If the controller allows a flow, it computes a route for the flow to take and adds an entry for that flow in each of the switches along the path. With all complex functions subsumed by the controller, switches simply manage flow tables whose entries can be populated only by the controller. A controller accomplishes this network programming via software and it is in this software that SDN's promise of flexibility comes in. The controller is a platform on which software is run, as well as being a communication gateway that software can communicate through. Most controller architectures are modular, allowing the controller to communicate with different kinds of devices using different methods as required.

SDN architecture is remarkably flexible: it can operate with different types of switches and at different protocol layers. SDN controllers and switches can be implemented for Ethernet switches (Layer 2), Internet routers (Layer 3), transport (Layer 4) switching, or application layer switching and routing. SDN relies on the common functions found on networking devices, which essentially involve forwarding packets based on some form of flow definition. It encapsulates and forwards the first packet of a flow to an SDN controller, enabling the controller to decide whether the flow should be added to the switch flow table. Switch forward incoming packets out; the appropriate port based on the flow table in which the flow table may include priority information dictated by the controller. Switch can drop packets on a particular flow temporarily or permanently as dictated by the controller.
SDN controller communicates with OpenFlow compatible switches using the OpenFlow protocol, running over the Secure Sockets Layer (SSL). Each switch connects to other OpenFlow switches and possibly to end-user devices that are the sources and destinations of packet flows. Within each switch, a series of tables typically implemented in hardware or firmware are used to manage the flows of packets through the switch.

Flow table tells switch how to process each data flow by associating an action with each flow table entry. Flow table consist of flow rules that guide the controller on action to be perform on a given particular packet. OpenFlow enabled device has an internal flowtable and a standardized interface to add and remove flow entries remotely (Naous, Erickson, Covington, Appenzeller and McKeown, 2008). Flow table is the basic building block of the logical switch architecture, each packet that enters a switch passes through one or more flow tables. Each flow table contains entries consisting of six components;Match Fields, Priority, Counters, Instructions, Timeouts and Cookie.

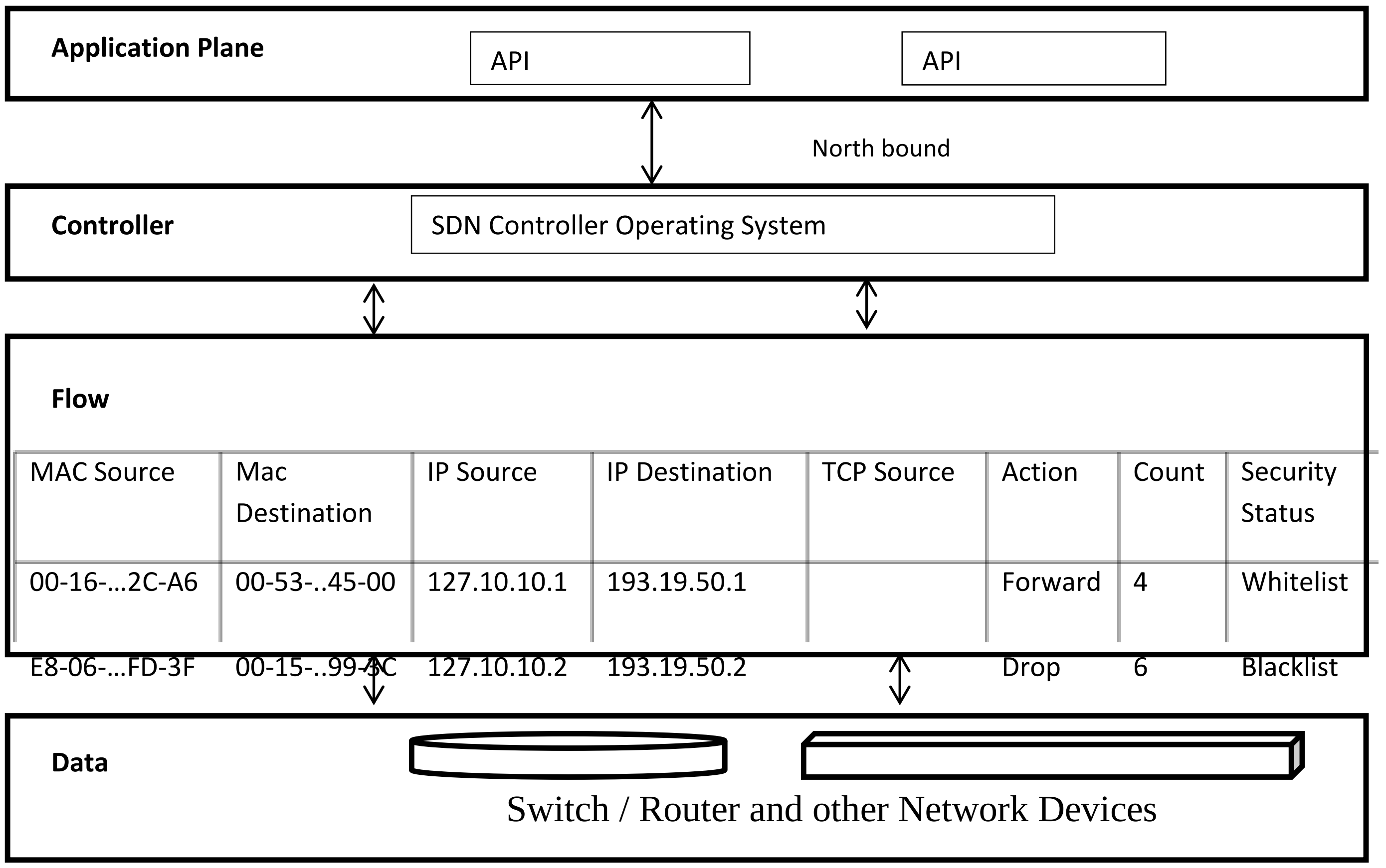
SDN switches are controlled by a Network Operating System (NOS) that collects information using the API and manipulates their forwarding plane, providing an abstract model of the network topology to the SDN controller hosting the applications. The controller can therefore exploit complete knowledge of the network to optimize flow management and support service user requirements of scalability and flexibility. Yutaka et. al. (2013) Propose a novel network system architecture that protects network devices from intra-LAN attacks by dynamically isolating infected devices using OpenFlow on detection. Seungwon et. al. (2013) had proposed an extension to the OpenFlow data plane called connection migration, which dramatically reduces the amount of data to-control-plane interactions that arise during the Inherent communication bottleneck that arises between the data plane and the control plane, which an adversary could exploit by mounting a control plane saturation attack that may disrupts network operations.

METHODOLOGY

To address the previously stated problem, the research provides secured SDN architecture concatenated with security control. Secured SDN paradigm where control plane check for the authentication of users' application through the API to confirm some security measure using the inbuilt white and black list for legitimacy confirmation of the users' application who are requesting to make use of control plane. If the application is from the black list it will be discarded but permitted if from the white list. A machine learning tool is used to update SDN architecture security black /white list using the packet flow movement on the table flow for updating. Figure 4 is the proposed SDN security architecture with the extension of control plane to contain some level of security control that interact with the proposed extended data plane flow table. Figure 5 contains extension of black/white list of the applications transactions; the extension will communicate with the control plane security control to supply the application security status to the flow table rule through the controller. The supplied security status is used by the controller to decide particular action(s) (forward/drop) to be taken on application requesting rule.

RESULT AND DISCUSSION

Research work comes up with a secured Software Defined Network (SDN) Security Architecture (figure 4) that identified the malicious source, and therefore prevents unauthorized access to the network by blocking packet from insecure source and automatically update (the grouping) white/black list. For subsequence packet(s) use, it compares and checks through its white/black list to identify the packet source and update the list using machine learning tool.


Application Plane
API
API
North bound
Controller
SDN Controller Operating System
Flow
Data
Switch / Router and other Network Devices

MAC Source	Mac Destination	IP Source	IP Destination	TCP Source	Action	Count	Security Status
00-16-…2C-A6	00-53-..45-00	127.10.10.1	193.19.50.1		Forward	4	Whitelist
E8-06-…FD-3F	00-15-..99-3C	127.10.10.2	193.19.50.2		Drop	6	Blacklist

Fig. 4: SDN Security Architecture

With extension of control plane and flow table with security features, a secured SDN architecture is designed (figure 6) where user's application is easily tested, and permit if not malicious attack while discard when suspected as suspicious. SDN is easily prevented from malicious attack, and made secured with some level of security control implementation into SDN Architecture that make it a secured SDN Architecture, this help to prevent malicious attacks by blocking packages from insecure source/networks. There is an extension of control plane called security control, this interact with extended aspect of flow table, consists of white and black list supplying it resolution based on the security status of incoming packet(s) to the secured SDN controller that will then place its security status on the flow table rule extension through the security controller. This will be used for decision making on the action to be perform on the said packet that is packet security status.

CONCLUSION

Security control system prevents malicious attacks from accessing SDN environment. Secured architecture promotes and encourages the openness of the SDN and prevent against its security challenges. The Security architecture for Software Defined Network (SDN) check security status of incoming packets populated by white/black list security controller for decision making on the arrived packed either to be permitted for transaction as legitimate packet or discarded as malicious packet.

REFERENCES

Bruce, H., Rossi, R. (2016). Software Defined Networking for Systems and Network Administration Programs. The USENIX Journal of Education in System Administration. 2(1). *www.usenix.org/jesa/0201*

Diego, K., Fernando, M. V., Ramos and Paulo, V. (2013). Towards Secure and DependableSoftware-Defined Networks. HotSDN'13, Hong Kong, China. ACM 978-1-4503-2178-5/13/08

Eddie, K., Robert, M., Benjie, C., John, J. and Kaashoek, M. F. (2000). The Click modular router. ACM Transactions on Computer Systems 18(3), pg. 263-297.

Furqan A., Iyad K., Ahmed S. A. (2013). New Networking Era: Software Defined Networking. International Journal of Advanced Research in Computer Science and Software Engineering. Volume 3, Issue 11, ISSN: 2277 128X. *www.ijarcsse.com.*

Haripriya, N., and Sangeethalakshmi, G. (2015). Evaluate Network Security and Measure Performance of Self Healing in 5G.Haripriya N et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (4), pgs. 3865-3870. ISSN:0975-9646. *www.ijcsit.com*

Mark, R., Marco, C., Arjun, G. and Nate, F. (2013). FatTire: Declarative Fault Tolerance for Software-Defined Networks. HotSDN'13, Hong Kong, China. ACM 978-1-4503-2178-5/13/08

McKeown, N., Anderson, T and Balakrishnan, H. (2008). OpenFlow: Enabling Innovation in Campus Networks. ACM SIGCOMM Computer Communication Review, 38(2), pg. 69-74

Mitchiner, M. M and Prasad, R. (2014). Software-Defined Networking and Network Programmability: Use Cases for Defense and Intelligence Communities. Cisco Public Information

Muhammad, H. R., Shyamala, C. S., Ali, N. and Bill, R. (2014). A Comparison of Software Defined Network (SDN) Implementation. 2nd International Workshop on Survivable and V Robust Optical Networks (IWSRON). Published by Elsevier B. Procedia Computer Science 32, pg1050–1055. *www.sciencedirect.com*

Naous, J., Erickson, D., Covington, G. A., Appenzeller, G. and McKeown, N. (2008). Implementing an OpenFlow Switch on the NetFPGA platform." ANCS '08, San Jose, CA, USA. ACM 978-1-60558-346-4/08/001

Raj, J. and Subharthi, P. (2013). Network Virtualization and Software Defined Networking for Cloud Computing: A Survey. Cloud Networking and Communications IEEE Communications Magazine Pp 24-31

Seungwon, S., Vinod, Y., Phillip, P. and Guofei, G. (2013). AVANT-GUARD: Scalable and Vigilant Switch FlowManagement in Software-Defined Networks. CCS'13, Berlin, Germany. ACM 978-1-4503-2477- 9/13/11. http://dx.doi.org/10.1145/2508859.2516684.

William, S. (2013). Software-Defined Networks and OpenFlow. The Internet Protocol Journal, Volume 16(1).

Wolfgang, B., and Michael, M. (2014). Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices. Future Internet 2014, 6, pg.302-336; doi:10.3390/fi6020302 ISSN 1999-5903. *www.mdpi.com/journal/futureinternet*.

Xitao, W., Yan, C. and Chengchen, H. (2013). Towards a Secure Controller Platform for OpenFlow Applications." HotSDN'13, Hong Kong, China. ACM 978-1-4503-2178-5/13/08.

Yutaka, J., Hung-Hsuan, H. and Kyoji, K. (2013). Dynamic Isolation of Network Devices Using Open Flow for Keeping LAN Secure from Intra-LAN Attack. 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems –

KES2013. Published by Elsevier B.V. Selection and peer-review under responsibility of KES International doi: 10.1016/j.procs.2013.09.163. Science Direct.22, pg. 810–819 1877-0509. *www.sciencedirect.com*