Volume-5, Issue-5, May - 2016 • ISSN No 2277 - 8160

Research Paper Engineering

Vulnerabilities of Openflow Network and Network Security for SDN Network

Joshi Maulik Y PG Student, Venus International College of Technology

Prof.Aniruddh Amin HOD, E&C Department, Venus International College of Technology

ABSTRACT *Software-Defined Networking (SDN) is unique architecture that is centrally manageable, very dynamic, low in cost, effective and adaptable. This architecture decouples the network functions as control plane and data plane functions. SDN is programmable and the underlying infrastructure to be abstracted for applications and network services. In this paper we provide a view on SDN/Openflow network, we will discuss the various types of vulnerabilities in the Openflow network and also the impact on SDN/Openflow controller. We also discussed on importance of Transport Layer Security (TLS) in Openflow network and resolve the issues which related to vulnerabilities in southbound interface.*

KEYWORDS : SDN, Control Plane, Data Plane, Security

INTRODUCTION :

Network intelligence is (logically) centralized in software-based SDN controllers, which maintain a global view of the network. As a result, the network appears to the applications and policy engines as a single, logical switch. With SDN, enterprises and carriers gain vendor independent control over the entire network from a single logical point, this greatly simplifies the network design and operation. SDN is described in article with the Open Networking Foundation (ONF) [1] definition: "In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications."

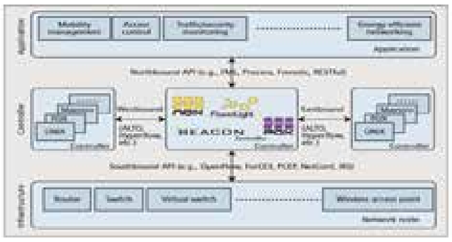

Fig. 1. SDN Functional architecture [2]

The future SDN architecture is described in Fig. 1. This architecture encompasses complete network platform [2].

The bottom tier of Fig. 1 involves physical network equipment including Ethernet switches and routers. This forms the data plane.

SDN mainly focuses on four key features [2]:

- Separation of the control plane and the data plane.
- Open interfaces between the control plane and the Data plane devices.
- A centralized controller and view of the network.
- Programmability of the network by external administration.

The focal level comprises of controllers that encourage setting up and tearing down streams and ways in the system. Controllers use data about limit and request, acquired from the systems administration hardware through which the activity streams. The focal level connections with the base level through an application programming interface (API) alluded to as the southbound API. Associations between controllers work with east and westward APIs. The controller to application interface is alluded to as the northbound API [2].

Useful applications, for example, vitality proficient systems administration, security observing, and get to control for operation and administration of the system are spoken to at the highest point of Fig. 1 highlighting the client control/administration partition from the information plane [2].

OPENFLOW SWITCH COMPONENTS

OpenFlow Switch which use openFlow protocol [5] and it is having 3 parts as shown in Fig.2: In SDN, OpenFlow Protocol is used as Communication Protocol between Communication devices of OpenFlow switches.

A flow table to indicate the switch as it has to process the flow. This flow chart is composed of actions. A Secure channel necessary to connect switch with a remote control device, for that TLS is used as secure channel. Third component is Openflow protocol. Using this protocol, Openflow provides a standard and open communication between the controller and the switch. An OpenFlow Switch consists of one or more flow tables and a group table, which perform packet lookups and forwarding, and an OpenFlow channel to an external controller [3].

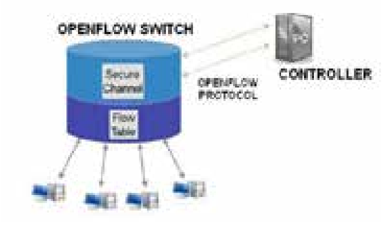

Fig. 2. Openflow Switch [3]

OPEN FLOW PROTOCOL

Openflow Standard [4] defines an Openflow protocol for communication between Openflow switch and the controller. The Openflow Switch must be able to establish communication with a controller at a user-configurable IP address, using a user-specified port. If the switch knows IP address of the controller, the switch initiates a standard TCP connection to the controller. Openflow protocol supports three messages types : controller-to-switch, asynchronous, and symmetric

- Controller/switch messages are initiated by the controller and may or may not require a response from the switch
- Asynchronous messages are sent without a controller soliciting them from a switch. Switches send asynchronous messages to controllers to denote a packet arrival or switch state change).
- Symmetric messages are sent without solicitation, in either direction (Hello, Echo, Error, and Experimental).

IV. SDN/OPENFLOW CONTROLLERS

SDN controller is the main device; it is responsible for maintaining all of the network rules and it distributes appropriate instructions for the network devices. In others words, the Openflow controller is responsible for determining how to handle packets without valid flow entries, and it manages switch flow table by adding and removing flow entries over the secure channel using Openflow protocol [3].

Controllers	Features
NOX	Multi-threaded C++-based controller written on top of Boost library.
POX	Single-threaded Python-based controller. fast prototyping network applications in research
Beacon	Multi-threaded Java-based controller that relies on OSGi and Spring frameworks.
Floodlight	Multi-threaded Java-based controller that uses Netty framework.
MuL	Multi-threaded C-based controller written on top of libevent and glib.
Ryu	Single-threaded Python-based controller that uses gevent wrapper of libevent.
Maestro	Multi-threaded Java-based controller.

TABLE I. SDN/Openflow Controllers

Classification SDN/OpenFlow Controller
Physical networks of devices, where equipment that forwards packets is actually located throughout the network

Software controllers, where software accesses and controls network services

Centralized and Distributed:
Centralized controller due to having one centralized controlling unit is beneficial for limited hosts but when the host increases, the centralized controllers face problem of bottleneck and it goes down so No scalability.

Distributed Controller can overcome bottleneck problem but they have to have some extra mechanism for supporting distributed characteristic, which controller will act as a master, what will be extra services of master controllers and all that, Example, Elasticon ,DISCO.

VULNERABILITY ANALYSIS
Controller Vulnerabilities
Attacks on and vulnerabilities in SDN controllers, are probably the most threats to SDNs. A hack or malicious controller could compromise the whole network. The use of a common intrusion detection system may not be enough, as it may be hard to find the proper combination of events that triggers a specific behavior and, importantly, to label it as malicious. Similarly, a malicious application can effectively do anything it changes in the network, since controllers only provide abstractions that translate into issuing configuration commands to the underlying infrastructure.

Solution: We have multiple several techniques can be used, such as replication (to detect, removal or mask abnormal behavior), employ the diversity (of controllers, protocols, programming languages, software images,), and recovery (periodically refreshing the system to a clean and reliable state). It is very important to secure all the sensitive elements inside the controller (e.g., crypto keys/secrets). Furthermore, security policies enforcing correct behavior might be mapped onto those techniques, restricting which interfaces an application can use and what kind of rules it can generate to program the network (along the lines of security-based prioritization).[8]

B. Control and Data plane communications Vulnerabilities
Using TLS/SSL, we can provide the security and it does not guarantee secure communication, and that compromises the controller–device link. Many more research papers report the issue of TLS/SSL communications and its major anchor of trust, the PKI infrastructure. The security of those communications is the weakest link in network, which could be a self-signed certificate, a compromised Certificate Authority, or vulnerable applications and libraries.

There are so many man-in- the-middle vulnerable implementations of SSL being used in critical systems. Moreover, the TLS/SSL model itself is not enough to establish and assure trust between controllers and switches. Once an attacker gains access to the control plane, it may be capable of take control of whole network to launch DDoS attacks. This lack of trust guarantees could even enable the creation of a virtual black hole network (e.g., by using Open Flow-based slicing techniques) allowing data leakages while the normal production traffic flows.

Solutions: we can use of oligarchic trust models with multiple trust-anchor certification authorities is a possibility. An-other is securing communication with threshold cryptography across controller replicas (where the switch will need at least *n* shares to get a valid controller message). Additionally, the use of dynamic, automated and assured device association mechanisms, one may consider, while in order to guarantying trust between the control plane and data plane devices.

VI. NEED OF TLS
Utilizing TLS has a higher specialized boundary for administrators to arrange it accurately, which incorporate the accompanying: producing a site wide declaration, creating controller endorsements, creating switch testaments, marking the authentications with the site wide private key,

We require one confirmation parameter when we do not use TLS in an operation to function, which may incentivize network administrators to skip TLS completely and rapidly evolving nature of Open Flow, many Switch and controller vendors have not fully implemented the specification and have skipped the TLS portion entirely. The lack of TLS support and lack of motivation to implement it leaves an avenue for attackers to infiltrate Openflow networks and remain largely undetected. [9]

Controller Name	Incorrect Message Length	Invalid Open flow Version	Incorrect Openflow Message type	Packet In Messa ge	Port Status Message
NOX	A	C	C	C	D
POX	B	B	C	D	D
Floodlight	B	C	C	C	D
Beacon	B	C	C	C	D
MuL	B	B	B	B	D
Maestro	A	C	C	C	D
Ryu	D	D	D	C	D

TABLE II. Security Analysis of different Controller under different Circumstances

A denotes controller crashed, B denotes controller closed the connection, C denotes controller processed the message without crashing or closing the connection, but the error was not detected, which is possible security vulnerability and D denotes controller successfully passed the test.

According to TABLE II, we found that mostly controllers are failed to detect malformed packets and forwarded them to network, so Openflow network will become very insecure and confirmed vulnerability in the openflow network.

VII.RELATED WORK
Security of SDN/Openflow controller is open issue and almost unexplored, presenting many challenges and opportunities. There are a few closely related works, specifically FRESCO. FRESCO [6] is an Openflow security application development framework designed to facilitate the rapid design, and modular composition of Openflow enabled detection and mitigation modules. Its main goal is to simplify development of security functions. Each FRESCO module includes five interfaces: input, output, event, parameter, and action. By simply assigning values to each interface and connecting necessary modules, a FRESCO [6] developer can replicate a range of essential security functions, such as

Firewalls, scan detectors, attack deflectors, or IDS detection logic. FRESCO has benefit of 90% reduction in lines of code.

If we compare it to standard implementations its resource controller component monitor switch status frequently and removes old flow rules to reclaim space for new flow rules, which will be enforced by FRESCO [6] applications.

Despite the success of openflow, developing and deploying complex openflow security services remains a significant challenge. The security of southbound interface; i.e. communication between control and data plane is important. In one type of man in the middle attack, if an attacker places a device between the controller and the switch to intercept openflow traffic, he/she could insert additional rules into the switch to record/modify sensitive traffic and gain access to protected segments of the network. If this communication is in plain text form then it is essential to provide some security mechanism. We have provided an approach for solving security related concern of southbound interface by assigning confidentiality between control and data plane communication. We have created a virtual environment, In Mininet [7] we used two OF switch, which was connected to OF controller and four hosts. We used beacon as a controller. Beacon controller gets packet from OF switch and in reply beacon will generate packet_out then beacon encrypts that packet_out and send this encrypted packet to reference switch.

VIII. CONCLUSION

Despite the success of Openflow, developing and deploying complex Openflow security services remain a significant challenge. In this paper we have provided an overview on SDN/Openflow Network; how does it work, its role of openflow controllers and most importantly, we discussed some vulnerability issues that were discussed in recent research work. In particular, security is a major issue and we debate on the role of TLS and acknowledged a problem that relates to malformed packet forwarding in controller, which can be a threat for the openflow network. We have implanted the referral system with beacon controller and two switches connected to it and we ensured the encrypted data flows in between. Here encryption is also not that much secured in today's world so in future we need to do more research on how more security provided in SDN Network

REFERENCES

[1] ONF white paper, "Software-Defined Networking: The New Norm for Networks," April 2012.

[2] Smeliansky R.L., "SDN for network security" Science and Technology Conference (MoNeTeC) IEEE, October 2014, pp. 1-5.

[3] R.Smeliansky, A. V. Shalimov, "Advanced Study of SDN/OpenFlow controllers," Proceedings of Eastern European Software Engineering Conference CEE-SECR, April 2013 10.1145/2556610.2556621, ISBN: 978-1-4503-2641-4.

[4] Kevin Benton, L. Jean Camp, Chris Small, "OpenFlow Vulnerability Assessment," Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking ACM, August 2013.

[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, 38(2):69–74, 2008.

[6] S. Shin, Porras, P., Yegneshwaran V., Fong, G. Gu, Tyson, "FRESCO: Modular Composable Security Services for Software-Defined Networks," In: Proceeding of the 20th Annual Network And Distributed System Security Symposium (NDSS), Jan. 2013.

[7] D. Turull, M. Hidell, P. Sjödin, "Performance evaluation of openflow controllers for network virtualization," 2014 IEEE 15th International Conference on High Performance Switching and Routing (HPSR), pp. 50-56, July.2014.

[8] OpenFlow Switch Specification Version 1.0 (Wire Protocol 0x01) Dec, 2009.

[9] https://www.opennetworking.org/sdn-resources/sdn-definition. Access on 07-06-2016.