Security in Software Defined Networking

[1]Shubham Mathur, [2]Priya Gupta,[3]Rohan Kumar Sachdev
School of Information Technology and Engineering
VIT University
Vellore, TN, India.
[1]shubham.mathur@engineer.com
[2]priyagupta.ynr@gmail.com
[3]rohan231094@gmail.com

***Abstract*— We live in a world where the things demand speed, accuracy and privacy. That's how internet came in. we have been using internet so much that we ourselves don't know how frequently we open the browser and start. A word we speak and we find internet. Internet is everywhere. We can easily connect more than one system to each other using the internet. Now we come to the problems we face in traditional networking. They are Static and inflexible networks. They are not useful for new business ventures. They possess little agility and flexibility. The traffic and maintenance was the major issue. This is where SDN (software defined networking) was born. Software-defined networking is a method to depict abstracting the network functions in order to control by software. This is done by disengaging the operating system (software) that makes decisions about where network traffic is sent from the underlying hardware that forwards traffic to the selected destinations. Hence maintaining the traffic and making the network flexible. But SDN do have some security issues as SDN has the control plane which controls the whole working of the network. For example, if attackers are able to take over the control plane or SDN controller they would essentially own the entire network and all its contents, possibly indefinitely. So in our research proposal we are identifying the new threats and the corresponding solutions to it.**

Keywords*— *Open Flow; intrusion detection systems; security; middle box; software-defined networking; authentication; encryption

I. INTRODUCTION

The undoubted growth in the technology has given birth to many new inventions. There were many issues with the traditional network, traditional networking is believed to be static but on the other hand dynamic networks are needed [1]. This thus makes the traditional networks harder and difficult to use and understand. [2].Networking and administrative policies are difficult to implement [3]. Hence this results in the problem with network management and thus the quality of the network suffers. To overcome these issues, it is important to shift to software-defined networks.

The change to software defined networks demands for separate data and control planes. The network's logic is implemented in the control plane, and the network switches are reduced to the elements forwarding it. [2] [3]. This provides a brief view of the network to the higher layers [3].We need something which not only provide the architecture but also the communication between in the control and the data plane. This is done by the Open Flow. Now open flow has become the standard for many industries.

Now this decoupling also makes the network vulnerable. [4] [5] [6]. These vulnerabilities bring great threat to the implementation of the software defined networking. Thus, it is important to rectify these security flaws in OpenFlow to ensure the progression to software-defined networks.

In this research proposal, we discuss the security concerns with OpenFlow architecture , and provide possible solutions to reduce them. Section II provides the idea of software defined networking and its layers. Section III discusses the methods to stimulate the vulnerability. Section IV flow rule data confidentiality.

II. SOFTWARE DEFINED NETWORK ARCHITECTURE

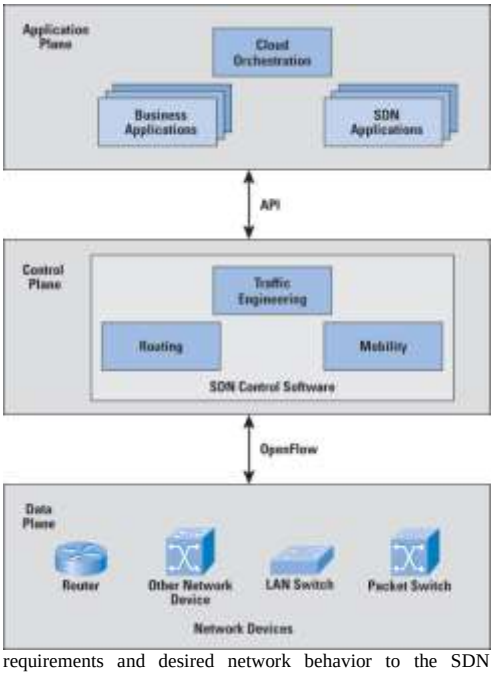
Switching function between a control plane and the data plane that are on separate devices are split in SDN approach. The data plane in SDN is responsible for packet forwarding , whereas in the control plane, it provides the "intelligence" in making or designing routes, routing policy, and setting priority parameters to meet QoE and QoS requirements and to manage with the continuous manage shifting .A software defined network can be reasonably divided into three planes: the first plane, which is the data plane, consists of network devices like switches and routers. The second plane is the controller plane and the third is the application plane. Communication within the SDN

architecture takes place between the switches and the controllers over the southbound interface, and between the controller and the applications over the northbound interface and between the controllers using east-west interface.

Fig. 1. The architecture of SDN

A. SDN Applications

SDN Applications are programs that explicitly, programmatically and directly communicate their network



requirements and desired network behavior to the SDN Controller via a **northbound interface (NBI)**. In addition, they may consume abstract view of the network for their internal decision making purposes. SDN Applications may themselves expose another layer of abstracted network control, thus offering one or more higher-level NBIs through respective NBI agents.

B. SDN Controller

The SDN Controller is a centralized/main entity in charge of translating the requirements from the SDN Application layer down to the SDN Data paths and providing the SDN Applications with an abstract view of the network (which may include statistics and events). An SDN Controller consists of one or more NBI Agents, the SDN Control Logic, and the Control to Data-Plane Interface (CDPI) driver.

C. SDN Data path/infrastructure

The SDN Data path is a logical network device that unveil visibility and uncontested control over its advertised data processing and forwarding capabilities. The logical representation may encompass all or a few of the physical substrate resources. SDN Data path comprises set of one or more traffic forwarding engines and zero or more traffic processing functions. These engines and functions may include simple forwarding between the data path's external interfaces or internal traffic processing or termination functions. One or more SDN Data paths may be contained in a single (physical) network element—an integrated physical combination of communications resources, managed as a unit. An SDN Data path may also be defined across multiple physical network elements.

D. Advantages of software defined networks over traditional networks

Due to the tight coupling between the data plane and the control plane the traditional networks are tedious. This forces introducing functionalities like load-balancing, deploying, intrusion detection and new protocols a hectic and long process. SDN simplifies network management by disengaging the data plane and control plane. The program logic in SDN is responsible to carry the functionalities in control plane, while the forwarding of frames and packets is done by the hardware components like switches and routers. The centralized logic hence introduces new feature and protocols into the network which is argument free. Another advantage of SDN is that new features can be tested without affecting the network before introducing them, therefore, minimizing the failures.

Centralization the control plane decision is useful in the better network management ensuring a centralized policy enforcement in SDN. One approach to achieve SDN is by adopting **OpenFlow** protocol for interaction between the control plane and data plane. OpenFlow is a standardized way of traffic management in switches by exchanging information between the switches and controller. Due to their ability to provide scalability, consistency and argument free management of large, heterogeneous networks, SDNs are becoming widely popular in the industrial sector.

II. SECURITY IN SOFTWARE DEFINED NETWORKING

Security issues related with SDN are due to the separation of the control and data planes that enables multi-tenancy and programmability, and which further introduces centralized/main management into the network architecture. The ability to share and dynamically operate the given physical network is one of the key.

Now SDN security issues relate to the new control plane model and to be precise it should importantly secure the inter components of the SDN. It seems that there is a lot of obstacles in security, which is accomplished by the SDN. SDN's programmability and its centralized management enables the greater chances to reduce any risks. Of all advantages there is a risk that the control plane gets hacked. The fact is control plane is an overview of the entire network and hence capable of reducing any risks dynamically.

However, while it is clear that centralization provides significant benefits, it also presents a number of challenges, like the fact that the sdnc is a highly attractive attack surface. Thankfully, resiliency, authentication, and authorization address this risk, reducing the impact of attack.

A. SDN vulnerabilities

In SDN, the controller is responsible in managing the functions of the control plane. Apart from several advantages this concept of centralization is prone to many security issues. Traditional networks bags the secure position because of their concrete hardware and network policies that were difficult to infiltrate. The paper discusses few vulnerabilities that software defined networks suffer.

Denial of Service (DoS): This is the most common attack that any network protocol suffers. The attacker searches for vulnerable hosts in the network, called zombies. Using the face of zombies by IP spoofing can inject a large volume of data traffic in the network to launch a Denial of Service (DoS) attack on OpenFlow switches.

The hardware devices in SDN due to decoupling of the planes are left open to be exploited like switches in the SDN architecture can be attacked by manipulating their behavior to disrupt network operations. The implementation we have done showcases this vulnerability.

Also lack of encryption over the south bound interface for communication and authentication of the switches at the controller makes the entire software-defined networking SDN network vulnerable. These loop holes in software-defined networking security can be used to gain the unauthorized access to the network. Multiple authorization certificates, enhanced cryptography techniques, with addition to dynamic flow control can be used to mitigate this threat.

A similar shortage of mechanisms to ensure secure the northbound interface which is used for communication between the controller and applications, can introduce additional threats. This can be avoided by giving network access to certain trusted applications or defining security guidelines for these applications.

Man-in-the-Middle Attack on SDN:In this paper, we examined the shortcomings encapsulating software-defined networking (SDN), particularly we have investigated the vulnerabilities of OpenDayLight SDN controller. Among all the possible software-defined networking vulnerable attacks, a man-in-the-middle attack using ARP poisoning was successfully launched to intercept the traffic between a client and the controller (OpenDayLight).

Details of the procedures, experiment method, and results are detailed in this report. This is a successful practical attempt to penetrate an SDN controller and be able to capture login credentials of the controller. The significance of this attack is severe as once the SDN controller is under the control of the striker, there will be no security at all for the entire network governed by this controller.

The theme behind a man-in-the-middle attack is that a foreign party intrudes in between the two end points of the communicating devices in the network. In this way, all the data which was meant to reach the destination point passes via the adversary. This allows the third party to gain access to confidential data such as login username and password. The layers of the OSI reference model are susceptible to a man-in-the-middle attack by several methods such as hijacking causing ARP poisoning or using network flaws to take the edge.

III. METHODOLOGY FOR SIMULATING VULNERABILITY

The simulation we performed has a controller being hosted in an isolated condition mimicking the real time scenario where it is less focused on security. The south bound interface between controller and switch is compromised when an adversary attacks in the middle. The attacker accesses the credentials intercepting the network.

Commented [SM1]:

Commented [PG2R1]:

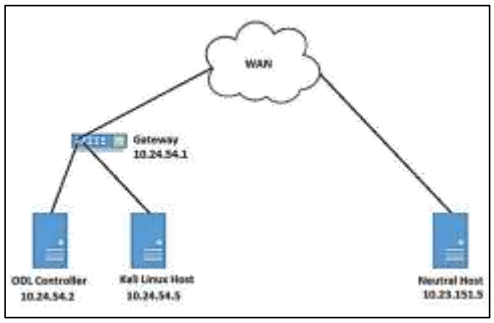
A. Experimental Setup

The experiment setup we have created is simulating three machines, the diagram describes the connection of these. On machine is running Ubuntu 16.04, hosting Open Daylight Controller (ODC), second machine is running Kali Linux and third machine running with same configuration as first one. We used Ettercap on Kali Linux machine which introduces the man-in-the middle attack between ODL controller machine and neutral host. The basic task of Ettercap is to perform ARP-poisoning attack on the Open Daylight Controller, thereby exploiting the user credentials consequently performing data traffic sniffing. The third machine is the neutral host which is hosting the Open Day Light controller trying to access the web interface. The gateway show in the diagram is provided by Virtual Box Host-Only Server.

Fig. 2. Experimental Setup

B. Performing Man-in-Middle attack

First Ettercap is configured to scan the network interface for port 8181 and 8080 for HTTP, as this is the most frequently used port and protocol by ODL. Then unified sniffing is selected in Ettercap under Sniff option. Network interface eth1



is selected which show all plugins, protocols, etc configured to be scanned for given network interface in Fig. 3

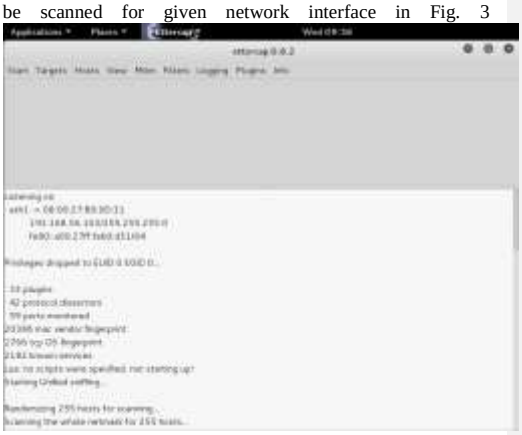


Fig. 3. Scanning hosts

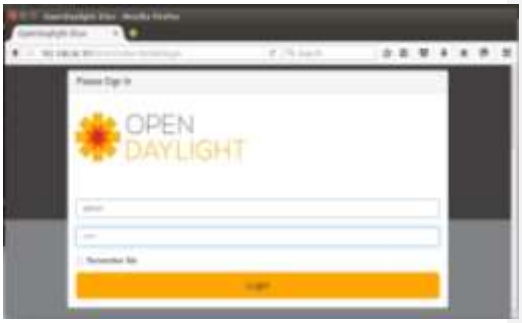


Fig. 4. Logging in ODL Dlux

Then hosts were scanned and list of hosts were generated. The ODL host is added to target 1and neutral host is added to target 2. Then ARP Poisoning is initiated to sniff remote connections. After that from neutral host ODL web interface, i.e. ODL Dlux is accessed from a web browser as shown in Fig.4. As the user enter its credentials to login the Ettercap sniffing the traffic sniffs it and shows the username and password in Fig .5. The username and password could be seen in Fig 5 as “admin”. To prevent ARP poisoning Snort can be used in ODL host. Snort is an intrusion prevention system (IPS),which was designed to test the multitude of the risks.. One such tool included in Snort is an ARP Spoof Preprocessor.

Fig. 5. Sniffing Traffic to get username and password

IV. ENSURING FLOW RULE DATA CONFIDENTIALITY IN SDN

The centralized controllers are an attractive target for attacks in the SDN architecture, since they are open to unauthorized access and exploitation. The OpenFlow protocol used for communication in the southbound interface is unencrypted and is susceptible to man in the middle attacks. A man-in-the-middle attack between the controller and switches is an ideal choice for attacking an SDN, as it can be used to intercept and tamper with the forwarding rules issued to the switch in order to gain control of network packet forwarding. After this has been achieved, further attacks can be implemented by attackers, such as black-hole attacks. In addition, we know that the controller and the switches maybe not be directly connected physically, i.e., a packet from a switch to the controller may travel through multiple other switches. Therefore, all switches and the hosts connected to them directly on the communication path are susceptible to be converted to agent nodes in a man-in-the middle attack

. In order to guard against man-in-the-middle attack, much research work has been done both in academia and in the industry. The most obvious approach is to create a secure channel between the controller and switches. A secure channel could be achieved by encrypting the data being transmitted on the south bound interface, by the implementation of TLS (Transport Layer Security) combined with authentication of the controller at the switch, may provide the necessary security and mitigate the threat to data confidentiality. Transport layer security manifests the asymmetric encryption, enforcing authentication and encryption using two protocols: the TLS Handshake Protocol and the TLS Record Protocol. The latter protocol uses an asymmetric algorithm for authentication and secure key exchange. The Record Protocol then uses this generated key as the input to a symmetric encryption algorithm to encrypt data between the two communicating devices, thus we can achieve encrypting of the data and rendering it unreadable to an interceptor.

Fig. 6. Transport Layer Security between Controller and Switch

TLS Protocol applies digital certificates to verify the identity of a host, these certificates embed the public key of an entity, along with other details such as the issuer name, subject name, validity period and the algorithm used for signing. The Certificate Authority generally issue and maintains these digital certificates. When a hosts requests for digital certificate the CA signs the certificate using its private key before sending it to the user; the user then decrypts the signed certificate using the public key of the CA. Thus, the user can verify that the certificate is valid and has not been tampered.

When a user wants to send any message to other user, it uses the public key of the other user and encrypts the message accordingly. Only the host which has the private key is capable to decrypt the message which is encrypted using its public key.

The switch initiates the communication by sending a First Packet to the controller. This First Packet indicates the encryption algorithm and the version of TLS that the switch supports. The controller then responds with its own packet; after which it proceeds to send its certificate to the switch. Thus, the switch can authenticate the controller. The switch then generates a private key and uses the public key of the controller to encrypt the private key and send it to the controller. Following this, the controller switches over to TLS thus authenticates the controller; however, this leaves the network open to attacks from compromised switches. To prevent a rogue switch from injecting malicious traffic, it is necessary to authenticate the switches in addition to the controller. This is done by authenticating the switches using their public key. Once the controller sends its certificate to a switch, it will then ask for the switch to verify its identity by sending a public key.

This prevents an adversary from adding malicious switches to the network, thus greatly increasing the security of the network.

CONCLUSION

Software Defined Networking is a paradigm which completely in its infant stage and is growing rapidly. The concept to make the network flexible and provide a dynamic nature to it, is a theme which envisages our future technology. SDN is a potential network technology which can meet the demands of enterprises including cost, maintenance and flexibility. However as every technology needs to be enhanced and improved, SDN also seeks issues to be worked on. In this manuscript we have tried to investigate few of the vulnerabilities of SDN, revolving mostly around the one associated with controllers. The man-in-the middle attack we simulated highlights the shortcomings on the security part. Through man-in-the middle attack it is possible to intercept the communicating devices and get the valuable credentials to compromise the controller. Consequently with a compromised controller network can be exploited, hence we can say that a system is secure as its weakest link. In this paper we made an attempt to mitigate attack by using the TLS, employing asymmetrical protocol. This security implementation provides encryption, authentication and detection of any infiltration. This reminds us, caution must be taken whenever a new component needs to be added to system. And the security should be considered when the system is designed, not an after-thought or a patch.

REFERENCES

[1] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K.Obraczka, and T.Turletti, 'A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks', *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, Feb. 2014

[2] K. Kirkpatrick, 'Software -defined networking', *Communications of the ACM*, vol. 56, no. 9, pp. 16–18, Sep. 2013.

[3] K. Cabaj, J. Wytrębowicz, S. Kukliński, P. Radziszewski, and K. T. Dinh, 'SDN Architecture Impact on Network Security', *2014 Federated Conference on Computer Science and Information Systems*, vol. 3, pp. 143–148, Sep. 2014.

[4] K. Benton, L. J. Camp, and C. Small, 'OpenFlow vulnerability assessment', *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, pp. 151–152, Aug. 2013.

[5] R. Kloti, V. Kotronis, and P. Smith, 'OpenFlow: A security analysis', *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pp. 1–6, Oct. 2013.

[6] D. Kreutz, F. M. V. Ramos, and P. Verissimo, 'Towards secure and dependable software-defined networks', *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, pp. 55–60, Aug. 2013.

[7] K. Cabaj et al., "SDN Architecture Impact on Network Security," in *Position papers of the 2014 Federated Conference on Computer Science and Information Systems*, Warsaw, Poland, 2014, pp. 143–148.

[8] Open Network Foundation. (April 13, 2012). Software-Defined Networking: The New Norm for Networks.

[9] Diego Kreutz, Fernando M. V. Ramos and Paulo Verissimo, "Towards Secure and Dependable Software-Defined Networks," in *HotSDN '13 Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, ACM SIGCOMM, NY, USA, 2013, pp. 55-60

[10] Antoine Feghali, Rima Kilany and Maroun Chamoun, "SDN Security Problems and Solutions Analysis," in International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS), Paris, France, 2015, pp. 1-5.

[11] Wen, Xitao, et al. "Towards a secure controller platform for openflow applications. Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM, 2013.

[12] Scarfone, Karen, and Peter Mell. "Guide to intrusion detection and prevention systems (idps)." NIST special publication 800.2007 (2007): 94.

[13] Sakir Sezer et al., "Are We Ready for SDN? Implementation Challenges for Software-Defined Networks," in IEEE Commun. Mag., vol. 51, issue 7, Jul 2013, pp. 36 – 43

[14] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, 'A security enforcement kernel for OpenFlow networks', Proceedings of the first workshop on Hot topics in software defined networks - HotSDN '12, pp. 121–126, Aug. 2012.

[15] Seugwon Shin et al., "FRESCO: Modular Composable Security Services for Software-Defined Networks," in NDSS Symposium, 2013, San Diego, CA

[16] Sandra Scott-Hayward, Gemma O'Callaghan and Sakir Sezer, "SDN Security: A Survey," in Workshop on Software Defined Networks for Future Networks and Service, Trento, Italy, 2013, pp. 1 – 7

[17] Xenofon Foukas, Mahesh K. Marina and Kimon Kontovasilis. Software Defined Networking Concepts [Online]. Available FTP: http://homepages.inf.ed.ac.uk/ Directory: mmarina/papers/ File: sdn-chapter.pdf

[18] R. Holz et al. "X.509 Forensics: Detecting and Localising the SSL/TLS Men-in-the-Middle". In: Computer Security. LNCS. 2012

[19] Vipul Gupta et al., "Performance Analysis of Elliptic Curve Cryptography for SSL," in WiSE '02 Proceedings of the 1st ACM workshop on Wireless security, NY, USA, 2002, pp. 87-94

[20] Kulkarni, Vikram, and Jayesh Kawli. "Analysis of OpenFlow Networks."

[21] Ramel, D. (2014). Explosive Growth Forecast for Software-Defined Networking. Virtualization Review. https://virtualizationreview.com/articles/2014/08/21/sdngrowth-forecast.aspx

[22] Hong, S., Xu, L., Wang, H., and Gu, G. (2015). Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures. NDSS 2015.

[23] Benton, K., Camp, L. J., & Small, C. (2013). Openflow vulnerability assessment. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (pp. 151-152). ACM.