Security Analysis for Enhancing Software Defined Networking

Kiruthikaa K V
Department of Computer Science and Engineering, Bannari Amman Institute of Technology, Sathyamangalam, Erode, Tamil Nadu, India.
kiruthikaasuresh@gmail.com

Abstract— **Software-defined networking (SDN) is a new approach to programmable networking in which network control is isolated from the hardware infrastructure. It provides a competing architecture that is used for designing and managing networks in an optimized manner. SDN delivers new applications and business services with a high speed and agility. Even though SDN has these remarkable advantages, it also refuge network security. In this paper, the role of SDN in addressing the emerging software security challenges is highy discussed.**

Keywords— **SDN, programmable networks, OpenFlow protocol, network functions virtualization (NFV), control plane, data plane, software vulnerabilities, security policy framework.**

I. Introduction

In the traditional approach to networking, the network functionality is defined by their physical structure i.e., the coupling between switches, routers and servers. In this type of networking, the required changes to the infrastructure can be done only with great complexity. So, it has to be enhanced in such a way that it could accommodate all the dynamic workload demands.

Nowadays, Most of the networking functionality are achieved through cloud environment. It is necessary to develop an architecture which supports cloud services with flexibility and agility. Software Defined Networks (SDN) provides an effective way to increase the efficiency of modern technologies and promises to allow networks to automatically change to meet application and business needs.

II. Architecture Of SDN

SDN allows applications to be aware of the network by taking a fresh approach to network architecture. In a traditional network, a network device like a router or switch contains both the control layer and the data layer. The control layer determines the route that traffic will take through the network, while the data layer is the part of the network that actually carries the traffic. SDN separates the control layer from the data layer to support virtualization [3].

SDN separates the network into three layers - application, control, and data as shown in Fig.1.

A. *Application Layer:* It consists of SDN applications, business applications or any network applications. These applications establish communication via an application programming interface (API). Developers write application programs to manipulate the logical network using this layer.

B. *Control Layer*: It is located between the application layer and the data layer i.e., it is logically centralized and separated from the data layer. It consists of the SDN controller that translates application requirements and manages the SDN data paths. The SDN controller is essentially a network operating system that constructs and presents a logical map of the network to services or applications that are implemented on top of it.

C. *Data Layer:* It consists of SDN data paths, which simplifies resource allocation and enables quality-of-service guarantees from end to end. The SDN data path is the logical network device that forwards the actual traffic.

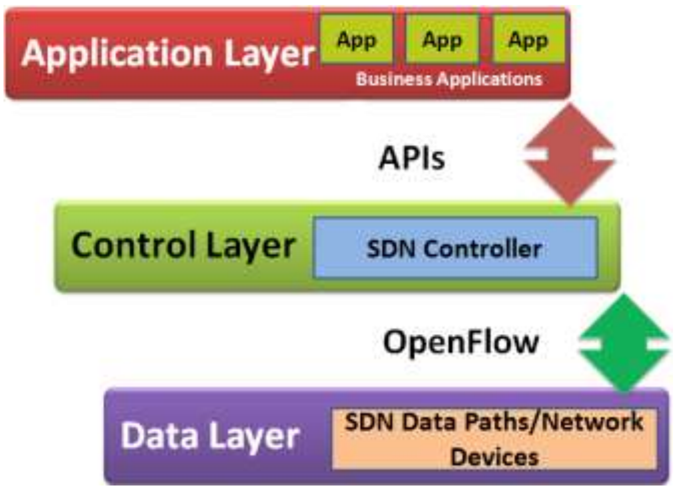

Fig.1 SDN Layers

III. Attacks On SDN Layers

Since SDN is a new paradigm to implement programmable networks via network virtualization, the probability of vulnerability will be higher. SDN may consist of hundreds of network segments, each of which requires a separate security mechanism.

Deployment of SDN varies by enterprises and network providers, which is considered as one of the main source of software vulnerabilities by attackers.

The OpenFlow protocol, developed by the Open Networking Foundation (ONF), is used to control the traffic flow of multiple switches from a controller and acts as an interface between the controller and the virtual network elements. This important characteristic of SDN also provides a path for the security attackers in such a virtualized atmosphere.

Centralized control or logically centralized control exposes a high-value asset to attackers. Attackers may attempt to manipulate the common network services or even control the entire network by tricking or compromising a controller. This is distinct from a larger number of autonomous assets in a completely distributed control domain.

Cloud services have created a diversified technology shift in the data centre. The future data centre is emerging as a highly virtualized environment that must address an assorted set of user needs. Even though protecting user data possess the principal importance, mobility and virtualization undergo new threats that must be managed and secured.

New types of threats arise due to the explicit programmatic access SDN offers to clients that are typically separate organizational or business entities. This new business model presents requirements that do not exist within closed administrative domains in terms of protecting system integrity, third-party data and open interfaces.

Control plane plays a centralized role in an SDN environment and hence the security policies must focus on protecting the control plane, and providing user authorization for network applications.

SDN uses southbound APIs to relay information to the switches and routers "below" whereas Northbound APIs are used to communicate with the applications and business logic "above."

Here we have discussed the possible risks which cause SDN to be a victim of security threats and attacks. The most common SDN security concerns include attacks at the SDN architecture layers.

A. Data Layer Attacks

The data layer is composed of network elements. So attackers may target the SDN infrastructure mainly by focussing on these network elements. The attacker can gain unauthorized access to the network and try to destabilize the network infrastructure. The attacker could negotiate a host that is already connected to the SDN network in order to perform attacks to subvert the infrastructure or network elements. Such activities on the data layer leads to Denial of Service (DoS) attacks or fuzzing attacks[1].

The centralized controller at the control layer makes use of southbound APIs and very new protocols to establish communication with the network elements. Each of these protocols has their own methods of securing the communications to network elements. However, many of these protocols may not have set them up in the most secure way possible.

The network attackers may involve in the activities like sniffing, spoofing, phishing, pharming, etc. Their ultimate aim would to be to hack or disrupt the flow of network traffic. So the network providers have to built-in some of the integrity mechanisms to network elements.

The attacker can sniff traffic on the computer network and perform a Man in the Middle (MITM) attack.

One solution is to use Transport Layer Security (TPS), which is a cryptographic protocol used to secure the communication over any computer network. It is an enhanced version of Secure Sockets Layer (SSL). It ensures integrity by providing authentication using X.509 certificates. As a special case, it provides better security for cloud based services. Organizations should prefer to use TLS to authenticate and encrypt traffic between network device agent and controller. Using TLS helps to authenticate controller and network devices/SDN agent and avoid eavesdropping and spoofed southbound communications.

B. Control Layer Attacks

SDN controller is present at the control layer, well-known as controller layer. For attackers, the ideal target is the controller. The attackers make use of spoofing to execute attacks over open flows. The attacker might instantiate new flows by either spoofing northbound API messages or spoofing southbound messages toward the network devices. If an attacker can successfully spoof flows from the legitimate controller then the attacker would have the ability to allow traffic to flow across the SDN at their will and possibly bypass policies that may be relied on for security.

Usually SDN controllers run on some form of Linux operating system. If the SDN controller runs on a general purpose operating system, then the vulnerabilities of that OS become vulnerabilities for the controller. Often times the controllers are deployed into production using the default passwords and no security settings configured. Sometimes the attacker might perform DoS attacks on the controller to make it fail.

The biggest of all, the attackers might create their own controller and acquire network access to send/receive flows such as a legitimate controller. The attacker could then create entries in the flow tables of the network elements and the SDN engineers would not have visibility to those flows from the perspective of the production controller. In this case, the attacker would have complete control of the network.

C. Application Layer Attacks

APIs are vulnerable to unintentional or intentional attacks and abuse including Distributed Denial-of-Service (DDoS), SQL and JavaScript or XPath Query attacks.

DDoS attacks have become one of the widespread cyber security threats. These attacks have a great impact over networks i.e., they bring down the computing, memory and network resources by leveraging huge amount of traffic over networks. This, in turn, either causes performance degradation or disruption of services to legitimate users.

Besides the conventional DDoS[2], now most of the organizations are challenged with application layer DDoS attacks. These attacks exploit legitimate HTTP requests to overwhelm target resources and also instigated mainly on web servers. These attacks are accomplished by establishing complete TCP connections with target servers using SYN flood DDoS. This type of attack exploits a part of the normal TCP three-way handshake to consume resources on the targeted server and cause it to be unresponsive. Essentially, with SYN flood DDoS, the offender sends TCP connection requests faster than the targeted machine can process them, causing network saturation.

In a SYN flood attack, the attacker sends repeated SYN packets to every port on the targeted server, often using a fake IP address. The server, unaware of the attack, receives multiple, apparently legitimate requests to establish communication. It responds to each attempt with a SYN-ACK packet from each open port.

The malicious client either does not send the expected ACK, or if the IP address is spoofed it never receives the SYN-ACK in the first place. Either way, the server under attack will wait for acknowledgement of its SYN-ACK packet for some time.

During this time, the application server cannot close down the connection by sending an RST packet, and the connection stays open. Before the connection can time out, another SYN packet will arrive. This leaves an increasingly large number of connections half-open. Eventually, as the server's connection overflow tables fill, service to legitimate clients will be denied, and the application server may even malfunction or crash.

IV. Security Analysis For SDN

When specifying a security mechanism for SDN networks, security dependencies between different components must be clarified. Multiple dependencies must be avoided.

The SDN Security is applied in the form of routing Protocols which include using certain Security measures such as MD5 for EIGRP, IS-IS, GTSM, Passwords and many others. Some administrators do not even apply these simple methods for securing standard IP Networks. If they apply the architecture of SDN with the same dispassion for Security, then there will be a lot of exposure to attacks at all three layers.

Compared with traditional networks, the separation of the control and data planes enables multi-tenancy and programmability, and introduces centralized management into the network architecture. In this new model, tenants run SDN apps that interface with the SDN controller, which sends instructions to network elements.

From a security perspective, the ability to share and dynamically operate the same physical network is one of the key security related differences between SDN and traditional architectures. As such, SDN security issues relate to the new control plane model, and more specifically to securing inter-component communication, and controlling the scope of applications and tenants through specific APIs and access policies. While it may sound like there are a number of obstacles to overcome, the programmability and centralized management brought about by SDN enables a much greater a level of autonomy to mitigate any security breaches by outweighing the need for additional technology.

In traditional networks, network elements tend to be monitored and managed individually. However, without the existence of standard protocols capable of interacting with all network elements irrespective of their vendor or generation, network management has become cumbersome.

The SDN approach enables coordinated monitoring and management of forwarding policies among distributed SDN resulting in a more flexible management process. While there is a risk of the SDN control plane becoming a bottleneck, the fact that it has an overview of the entire network, makes it capable of mitigating any reported incident dynamically. For example, a DDoS attack can be detected and quickly mitigated by isolating the suspect traffic, networks or hosts. Unlike traditional DDoS appliances, which generally carry only a local view of the network and the centralized elements possess a much broader view of network topology and performance, making the SDN an ideal candidate for the dynamic enforcement of a coherent security posture.

The vital nature of the SDN controller states that additional security measures are needed to be taken to protect it from the offenders. As a must thing, network traffic must be highly protected to prevent tampering. The basis for effective security is the ability to uniquely identify all components and users of a system and verify identities with a trusted source. Without a strong identity framework, the ability to build effective authentication, authorization, and accounting implementations will be limited.

Authentication and authorization are the processes used to identify an unknown source and then determine its access privileges. Implemented correctly, these processes can protect networks from all the certain types of attacks.

Encryption can also be used to prevent control data from being leaked. But, it is also not sufficient to protect against man-in-the-middle-type attacks. So, all communication within the control plane must be mutually authenticated. Security protocols like TLS, IPSec and SSH provides a means for mutual authentication as well as for replay attack protection, confidentiality, and integrity protection.

Mutual authentication involves some difficulties such as how to bootstrap security into the system. One way to solve this is by using SSL certificates. Every SSL certificate that is issued for a CA-verified entity is issued for a specific server and website domain. When a person uses their browser to navigate to the address of a website with an SSL certificate, an SSL handshake occurs between the browser and server. Information is requested from the server, which is then made visible to the person in their browser window. Once the secure session has been initiated, a trust seal will appear. This trust seal will contain additional information such as the validity period of the SSL certificate, the domain secured, the type of SSL certificate, and the issuing CA.

The SDN controller provides network configuration information through API calls to its services, which enables tenants to use SDN applications to control network behaviour. This situation is somewhat alarming, given that physical hardware resources may be shared among rival tenants. While ordinary security measures such as argument sanitization and validation must be in place, the SDN controller also needs a solid authentication, authorization and accountability infrastructure to protect the network from unauthorized changes. Strong authentication and authorization provides additional protection, as it prevents an attacker from impersonating an SDN component, especially the SDN controller.

For networks built using SDN techniques, it is possible for the same physical network to be shared among several tenants, which can in turn manage their own virtual networks. Multi-tenancy allows for better utilization of network resources, lowering the total cost of ownership. For tenants, SDN shortens the time taken to react to changing situations through automatic scaling of resources. To maintain an acceptable level of security, tenants should not be able to interfere with each other's networks, and need not even be aware that they are sharing network resources with others. Tenant isolation is an important feature of SDN framework security.

Analysis of the SDN architecture identifies numerous means for elements inside the system's trust boundary to compromise the availability of the logically centralized control. Strong authentication based on assured identity is, therefore, critical to the security of the system. There are several use cases for which elements external to the SDN system like network applications will require access to a subset of system resources through defined interfaces. For such circumstances, access control mechanisms with various privilege levels should be employed to authorize external parties and authenticate their access to the system such as role-based access control [12].

Role-based access control (RBAC) is a commonly used approach for restricting the actions permitted by an application by assigning a role to it. Roles can be defined on a host, user or application basis. In effect, RBAC is a security policy enforcing system. The fewer the number of permitted actions, the more limited the exploitable functionality. When implemented correctly, RBAC can be invaluable. Unfortunately, this approach is rather cumbersome in systems with very narrowly defined roles where frequent changes take place.

At the other end of the scale, RBAC loses its edge if roles are too loosely defined. For the purposes of system integrity assurance, every event that occurs in the system should be recorded in a log. How these logs are stored and secured against improper access also needs to be considered, and an external host is recommended. During communications, the identity of a device can be indicated explicitly by the information (e.g., identifiers, credentials, IP addresses, etc.) transferred with the packets, or implicitly by the key used to secure the packets.

A. *Securing Data Layer*

Existing SDN architecture uses X86 Processors and TLS for securing the data layers. The long-term HTTP Sessions are vulnerable to an extensive variety of attacks that could highly expose the integrity of the Data Plane. This could cause Cloud-based Services to be compromised. Companies should use TLS or SSH to provide Authentication as well as Encryption for the traffic between Network Device and Controller.

The proposed security mechanism for securing data layer of SDN architecture comprises of encrypting legitimate HTTP sessions using Digital Signature Algorithm (DSA). The digital signatures generated are used to validate the network traffic between tenants and the hosts on the data plane. These are also used to authenticate the hardware resources and authorize the corresponding access methodologies.

Digital signatures are generated through DSA, as well as verified. Signatures are generated in conjunction with the use of a private key and the verification takes place in reference to a corresponding public key. Each tenant has their own paired public and private keys. Because a signature can only be generated by an authorized person using their private key, the corresponding public key can be used by anyone to verify the signature.

Since SDN accomplishes the network virtualization and OpenFlow concept, the chances of network based attacks are on the mark. Data plane isolation need to be resolved by implementing centralized management which also enhances the security of data layer.

B. *Securing Control Layer*

Programmability is a double-edged sword; it offers flexibility to implement newly innovated market-driven applications but it also opens the door to malicious and vulnerable applications. Authentication and different authorization levels should be enforced at the point of application registration to the controller in order to limit the controller exposure.

The Controller is the primary target for the hacker's attacks and thus, it must be secured. Securing the SDN controller and the network elements directly depends on the security mechanisms to protect the operating system. So the operating system must be secured using cryptographic algorithms and the controllers must be watched for any suspicious activities now and then when it receives/sends any request/reply respectively.

The unauthorized access to SDN control system must be prevented. SDN Frameworks should allow access to configuration panel of the controller for authenticated administrators. RBAC may be required for controller administrators in order to make it secure. Logging and audit trails could be used for checking unauthorized access by controller administrators.

High-Availability (HA) controller architectures can be used if there is a chance of DoS attacks at the control layer. SDNs that use redundant controllers could suffer the loss of a controller and continue to function. This would raise the bar for an offender trying to execute DoS on all the controllers in the system. But, such attacks would not be particularly stealthy.

C. *Securing Application Layer*

APIs are also key target for attacks since these APIs use Java, JSON, C, XML, REST and Python, and some, such that offenders could gain control of the SDN infrastructure by exploiting vulnerabilities in any of these. Leaving a default password on such APIs would allow a hacker to easily guess it and then create packets to forward on to a controller's management interface to determine the structure of the SDN network or even set up their own one. It is also mandatory to authenticate an application's access to the control plane and prevent this authenticated application from being hacked.

APIs can be coded securely using TLS or SSH. Also, the authentication and encryption methods should be deployed on all communications between applications and services requesting SDN services and data, and the controller serving these requests.

V. Conclusions

Software defined networking is on the success road of network management. But, the security issues raise hands as bottlenecks to its enhancement. This paper provides a detailed analysis over the various attacks and risks over the SDN architecture and openFlow protocols. While designing SDN architecture for any application, the discussed hurdles can be overwhelmed in order to implement a threaten-free network using modern virtualization techniques. Such architectural designs improve the transition of existing networks to SDN.

References

[1] Haopei Wang, Lei Xu, and Guofei Gu. FloodGuard: “A DoS Attack Prevention Extension in Software-Defined Networks”, in DSN’15, 2015.

[2] N. Iyengar, G. Ganapathy, P. Kumar and A. Abraham, “A multilevel thrust filtration defending mechanism against DDoS attacks in cloud computing environment”. International Journal of Grid and Utility Computing, 2014, vol. 5(4): pp. 236-248.

[3] Nunes, B., et al., “A survey of software-defined networking: Past, present, and future of programmable networks”, Communications Surveys & Tutorials, IEEE, 2014. 16(3): p. 1617-1634.

[4] Mi, X., et al., “NO stack: A SDN-based framework for future cellular Networks”, IEEE, 2014.

[5] N. L. M. v. Adrichem, B. J. v. Asten and F. A. Kuipers, “Fast Recovery in Software-Defined Networks,” Third European Workshop on Software Defined Networks, 2014, pp. 61-66.

[6] L. F. Muller et al.,“Survivor: An enhanced controller placement strategy for improving SDN survivability,” , IEEE Global Telecommunications Conference (GLOBECOM), 2014.

[7] G. Garg and R. Garg, “Review on architecture and security issues in SDN”, in International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE), Vol. 2, Issue 11, pp. 6519-6524, 2014.

[8] Kim, H. and N. Feamster, “Improving network management with software defined networking”, Communications Magazine, IEEE, 2013, p. 114-119.

[9] Seungwon Shin and Guofei Gu,“Attacking Software-Defined Networks: A First Feasibility Study”, in Proceedings of ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN’13), 2013.

[10] Jain, Sushant, et al., “B4: Experience with a globally-deployed software defined WAN”, ACM SIGCOMM Computer Communication Review, Vol. 43. No. 4. ACM, 2013.

[11] Diego Kreutz, Fernando M. V. Ramos, and Paulo Verissimo.“Towards Secure and Dependable Software-Defined Networks”, in Proceedings of ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN’13), 2013.

[12] Xuebin Chen , Shufen Zhang and Dianchuan Jin, “The applications and research of RBAC in Network Collaborative Environment”, in 3rd International Conference on Information Sciences and Interaction Sciences (ICIS), IEEE, 2010.

[13] N. McKeown, T. Anderson, H. Balakrishnan, et al. “Openflow: enabling innovation in campus networks”, ACM SIGCOMM Computer Communication Review, 2008, vol. 38(2), pp. 69-74.