Study of a security exploit in an SDN context

Francois Richard
Vigo, Spain

Abstract

In this paper, we will talk about the SDN technology. Explain the core idea behind it before presenting several security problem in it. We will show how easy it is currently to create a single point failure on the SDN-based network. Finally we will have a reminder of how a few good practices coupled with some specific solutions can improve the security in a SDN-based network even though a lot is still to be done.

Keywords: Software-defined-networking, OpenFlow, security

1. Introduction

Software Defined Networking (SDN) is a new paradigm in networking that introduces a change in the way network configuration and real-time traffic management is done. The main goal of SDN is to render the network management more flexible compared to the legacy network architecture. To fulfill this objective, SDN separates the control plane from the forwarding plane and enable one control plane to controls multiple devices. The end goal of doing that is to render the data plane completely programmable and to solve problems due to the hardware component. In SDN a centralized control device configures and control dynamically all the networking devices of the network. This high level of control results in a better management of the resources of the network. However, despite its great potential SDN is not exempt from problems and is encountering multiple challenges in practical implementation.

In this paper we will first present the technology that is SDN in more details. Once that is done we will focus on the security problem that exists currently and develop one of them by demonstrating how easy it is to exploit it to impact gravely the network behind the controller. Finally, the last section will present solutions and good practices that can improve the SDN security and mitigate some of the threats that we will evocate here.

2. SDN Architecture

As said before, the basic idea behind SDN is to separate the control plane from the data plane.By doing that, it permits us to separate the processes like configuration, resources allocation, and traffic management in three separate layers: application layer, a control layer and data layer that can communicate with the adjacent plane thanks to API.

The application layer or plane contains all the network and business applications. It has a very basic view of the network organization transmitted from the Northbound API. Once the applications receive this overview of the network, they communicate their requirements to the control plane.

The control plane: he is responsible for the decision concerning the gestion of the traffic on the network. His main component is the SDN controller. This controller will change the rules for traffic forwarding to respond to the need of the application of the application plane. Those rules are then transmitted to the data plane.

The data plane: the data plane is a set of network devices like switches, routers, firewall and so forth. The only job of the data plane is to forward the packet in an efficient way according to the traffic rules he received from the control plane. Those rules are transmitted from one plane to the other by the Southbound API. Currently, the most used SouthBound protocol is OpenFlow.

2.1. Southbound

Of course, as SDN is still developing, OpenFlow is not the only South-Bound protocol existing. Other protocols that could be used are Extensible Messaging and Presence Protocol (XMPP) QND Cisco OpFlex.

XMMP: share some similarities with the SMTP protocol and is defined in an open standard and follows open systems development to allow interoperability but it is not exempt from weaknesses. It does not guarantee a QoS for the exchange of messages between the XMPP client and server and doesnt allow end-to-end encryption which is a basic security requirement in modern architecture.

Cisco OpFlex: is another Southbound protocol aiming to become a standard policy implementing the language. Contrary to OpenFlow which centralizes all network control functions on the SDN controller, Cisco OpFlex focuses on implementing and defining policies. The reason for that is to avoid a scenario where the controller scalability and control channel communication from becoming the network bottleneck by leaving some level of decision to the devices by using some legacy protocols. The allows bidirectional communication of policies and networking event as well as monitoring information. This information can then be used to make some adjustments in the network configuration. One of the main advantages of Cisco OpFlx is that it offers a great level of control to the developer for networking control applications and dont rely too much on network vendors.

2.2. Northbound

The role of the Northbound protocol is to act as the connection between the application and the control plane. It is a very important part of SDN and as such, it has been actively developed by numerous vendor with the aim of reducing the cost of future IT installation.

Representational State Transfer: or REST follows the software architecture style developed for World Wide Web. The client and the server can be developed at the same or separately and can be from different vendors as well. It is prevalent in a lot of controller architecture as the northbound interface to use with Java API. One of the major drawbacks is that it doesnt show recent event on the network. If a change happened it is necessary to refresh to get the information.

Open Services Gateway Initiative (OSGi): it is a set of specification for dynamic application composition using Java bundles. Those bundles can be installed and updated by a lifecycle API. Among the SDN controller that use OSGi is the OpenDaylight project. OSGi allows the starting, stopping, loading, unloading of Java-based network module without a need to stop then restart the controller.

Intent-Based Approach: this northbound interface allows the application to describe their intent instead of the methods to achieve it. this allows the user or the application to only specify the intent and the SDN controller translate it into low-level configuration commands in the data plane for execution. In this case, the application or the user is not aware of the infrastructure configuration and that allows for more flexibility and automation during application development and the deployment of services. Intent-based

Northbound Interface (NBI) is very appealing because it has the following advantages:

- application scalability, the application developer doesnt need any knowledge about the actual network and can concentrate on the application instead
- great portability: As it doesnt need any specifics about the actual network it shows great portability
- Intent-Based directives to the SN help to form a cohesive device configuration to avoid resource conflict
- The management of the network is simplified compared to low-level network resource managing.

However, currently, this solution is not fully developed as it needs a language to translate the user or application intent and multiple projects are in development.

3. Security problems in SDN

Software-Defined Networking (SDN) is an approach to networking that separates the control plane from the forwarding plane to support virtualization. SDN is a new paradigm for network virtualization. Most SDN architecture models have three layers: a lower layer of SDN-capable network devices, a middle layer of SDN controller(s), and a higher layer that includes the applications and services that request or configure the SDN. Even though many SDN systems are relatively new and SDN is still in the early phase of development, once it is widely deployed it will be the target of a lot of attacks. We can anticipate several attack vectors on SDN systems. The more common SDN security concerns include attacks at the various SDN architecture layers.

3.1. Security of the Southbound

The first part at risk the network itself o more precisely the elements inside of it. An attacker could theoretically gain unauthorized physical or virtual access to the network or compromise a host that is already connected to the SDN and then try to perform attacks to destabilize the network elements.

This could be a type of Denial of Service (DoS) attack or it could be a type of fuzzing attack to try to attack the network elements.

There are numerous southbound APIs and protocols used for the controller to communicate with the network elements. These SDN southbound communications could use OpenFlow (OF), Open vSwitch Database Management Protocol (OVSDB), Path Computation Element Communication Protocol (PCEP), Interface to the Routing System (I2RS), BGP-LS, OpenStack Neutron, Open Management Infrastructure (OMI), Puppet, Chef, Diameter, Radius, NETCONF, Extensible Messaging and Presence Protocol (XMPP), Locator/ID Separation Protocol (LISP), Simple Network Management Protocol (SNMP), CLI, Embedded Event Manager (EEM), Cisco onePK, Application Centric Infrastructure (ACI), Opflex, among others. Each of these protocols has their own methods of securing the communications to network elements. However, many of these protocols are very new and implementers may not have set them up in the most secure way possible. The most used protocols for Southbound API right now are mainly OpenFlow and OpenvSwitch Database Management Protocol that are developed by the Open Networking Foundation.

Another possibility for an attacker is to exploit those protocols and attempt to instantiate new flows into the devices flow-table. The attacker would want to try to spoof new flows to permit specific types of traffic that should be disallowed across the network. If an attacker could create a flow that bypasses the traffic steering that guides traffic through a firewall the attacker would have a decided advantage. If the attacker can steer traffic in their direction, they may try to leverage that capability to sniff traffic and perform a Man in the Middle (MITM) attack.

Furthermore, if an attacker gains access to a switch on the network, he could eavesdrop on flows to see what flows are in use and what traffic is being permitted across the network. From there it is very easy for the attacker to listen to the communication on the southbound interfaces between the switch and the controller. This information can later be used to perpetrate another attack or simply for reconnaissance purposes.

Many SDN systems are deployed within data centers and data centers are more frequently using Data Center Interconnect (DCI) protocols such as Network Virtualization using Generic Routing Encapsulation (NVGRE), Stateless Transport Tunneling (STT), Virtual Extensible LAN (VXLAN), Cisco Overlay Transport Virtualization (OTV), Layer 2 Multi-Path (L2MP), TRILL-based protocols (Cisco FabricPath, Juniper QFabric, Brocade VCS

Fabric), Shortest Path Bridging (SPB), among others. These protocols may lack authentication and any form of encryption to secure the packet contents. These new protocols could possess vulnerabilities due to an aspect of the protocol design or the way the vendor or customer has implemented the protocol. An attacker could be motivated to create spoofed traffic in such a way that it traverses the DCI links or to create a DoS attack of the DCI connections.

3.2. Security of the Controller

It is obvious that the SDN controller is an attack target. An attacker would try to target the SDN controller for several purposes. The attacker would want to 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 controller, it basically means that he got a clear view of the network and can bypass any security measure put in place. This is one of the biggest problems with the SDN architecture.

Most of the time, SDN controllers are 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. Usually, the controllers are deployed into production with only the default password and no other security settings configured. The fear of creating a malfunction after finally getting it to a working state end up being a problem because a lot of security exploits are left unattended.

Lastly, it would be bad if an attacker created their own controller and got network elements to believe flows from the rogue controller. The attacker could then create entries in the flow tables of the network elements and the network administrator would not have any visibility on those flows from the perspective of the real SDN controller. In this case, the attacker would have complete control of the network.

3.3. Security of the Northbound

Attacking the security of the northbound protocol would also be a likely vector. There are many northbound APIs that can be used by SDN controllers. Northbound APIs can use Python, Java, C, REST, XML, JSON, among others. If the attacker is able to leverage the vulnerable northbound API, then the attacker would have access to the controller and through the controller, hell have access to the whole network. If the controller lacked any form of security for the northbound API, then the attacker might be able to create their own SDN policies and thus gain control of the SDN environment.

Once again, most of the times the password set for the Northbound API is not very secure and easy to figure out after some time. If an SDN deployment dont change this default password and the attacker can create packets via the controllers management interface, then he can also query the configuration of the SDN environment and put in his own configuration.

4. Failure of Southbound security

In this part of the paper, we will see how a well executed DoS attack from the southbound can disastrous consequences for the network. However, the aim of this DoS attack will not be a switch like it could be in a Legacy network. Since the study take place in an SDN context, this attack will be targeted toward the controller to put some light on one of its vulnerabilities.

First, a quick reminder, in a normal functionment when the network is set up, the controller sends all the flows to the switches. The switch then treats the packets on the network according to the flows that were transmitted by the controller and that form his flow table. However, it is possible that at some point the switch receives a packet that doesnt fit any of the rules of the flow in the flow table. When faced with such a packet, the default behavior is to inform the controller by sending a packet that is the packet-in. When the controller receives the packet-in, he can decide to do two different things. The first option is to create a new flow rule that fit the packet and that will serve as a reference for all future packet fitting those criteria. The other option is to decide to drop the packet. No matter the solution chosen, in both cases, the controller will then send a packet-out to answer the query of the switch.

In this case, the idea is to generate a lot of packets that would not appear suspicious but that cannot be handled by standard flow rules passed down by the controller. Indeed, generating a large number of such packet will result in a great number of packet-in to be sent to the controller. This will, in turn, reduce the resources available for the controller to deal with other problem on the network and the traffic generated will also impact the fluidity of the network as a whole. Another interesting point of such an attack is that it really impact directly the controller. Since what actually reach the controller are packet-in send by the switch, it will reach it even if some secure protocols like TLS are configured for the exchange between the switch and the controller.

4.1. the testbed

To realize this test, we used virtual machines to host Mininet to emulate the network, a floodlight controller and a packet crafting tool to prepare the attack.

4.1.1. Mininet

Mininet is a network simulation tool that permit to emulate SDN-based network. Using this solution we emulated a small network with 3 hosts and 1 switch. to that that we add 2 other machines that will serve as hosts and will be used to start the attack and an external controller.

4.1.2. Floodlight

The controller used in this experience is a floodlight controller. The Floodlight Open SDN Controller is an enterprise-class, Apache-licensed, Java-based OpenFlow Controller. It is supported by a community of developers including a number of engineers from Big Switch Networks. It also whas the adavantage of being easy and quick to install on a machine, got a web-UI available to get a quick overview of the network and it is described as being designed for high performance.

4.1.3. Scapy

Scapy is a packet crafting tool that gives a lot of freedom. In most networking tool, the information available to the user is only the information that the creator of the tool needed at the time. Scapy is functioning in a different way. It is a packet crafting tool that let the user create networking packet in the way he wants. With this tool, it is possible to stack networking protocols in the order that we want.

It also differs from other networking tools in that it gives us the raw information about what is happening with the packet we send on the network. There is no processing by the computer to give a more readable output to the information. The feedback is complete so that no information end up being overlooked and lost for the user. However, the consequence of this way of doing things is that it makes things more difficult for the user as it is not really friendly towards beginners. This solution is available for Windows and Linux and was used to craft specific packet for the attack.

4.2. The attack

The aim of the attack is to generate a lot of packets that don't fit any of the existing flows in the flow table of the switch. when the switch receives a packet like that, he compile information about it and send it to the controller in packet-in. the role of this packet-in is to ask the controller to define a behavior to adopt with this kind of packets. The controller will then either defines a new flow or decide that such packets should be dropped. he then send his answer to the switch via packet-out.

By creating a script that repeatedly craft then send the packets on the network we generate a lot of traffic. All this traffic result in an exchange of packet-in and packet-out between the controller and the switch. the result of the test show that the controller is very quickly overwhelmed. it will first retrograde to an older version of the protocol using less resources for the algorithm. Finally it run out of resources and can't handle new packets and sever the connection with the switch.

4.3. Consequences

After the controller shut down due to lack of resources and a rising number of error, we can observe severe repercussion on the network. Some simple test on the network like iperf and pings were all non-conclusive, showing an impossibility for any machines on the network to communicate with each other. From this we can safely deprive that the cause of this loss of connection is due to the loss of the controller.

5. Way to improve security

From this test and how easy it was to put in action, we have shown that SDN is far from being completely secure. However, a lot of researches are done currently to find ways to mitigate the threats on the network in an SDN context.

5.1. the security on Openflow

As we have seen earlier, SDN is far from being completely secure and can still be improved on a lot of points. In this attack, we exploited a default in the way the Southbound works and particularly the way the OpenFlow protocol operates. We will now present the potential ways to improve the security of this protocol.

5.1.1. *Coping with single point failure*

In the legacy network, the network intelligence is distributed throughout the whole network between the different machines with the routing tables and forwarding tables of the routers. In an SDN environment, all those information are centralized at the controller. This creates a potential single-point failure where if the controller fails the whole network is compromised. This is exactly what happened in the attack that was done earlier. To deal with this problem it thus becomes mandatory to implement multiple controllers for the network to ensure some sort of redundancy.

5.1.2. *Prevent disclosure of information*

OpenFlow replaces the old routing and forwarding tables in the switch with flow tables. Those tables contain rules that are matched to flow defined by the controller. Compared to legacy network, the controller doesnt see the network packet by packet but flows on them. This is called flow aggregation and permit to have a globalized view of the network. The problem of this way of doing things is that it is much too open to a man in the middle attack. Indeed, if someone starts listening to the communication between a switch and a controller he could quickly figure out the different flows defined on the network. From this point, he would be able to gain insight on the various part of the network to then formulate an attack. Thankfully it is possible to make it more difficult for the listener by associating one flow with multiple flow rules. This makes it more difficult to interpret the flow and as such offer some protection to the machines generating it.

5.1.3. *Mitigate DoS attack*

As we have seen previously, the DoS attack in an SDN environment can have disastrous consequences for the network due to the centralization of information. Thats why it is very important to find solutions to deal with this type of attack. One of the ways to do it would be to limit the flow-rate of the packets to ensure that the controller is never flooded with packets. However, this would also impact the quality of services of the whole network as it would require to slow down the whole network. Other solutions that are researched right now are Flow-based IDS (Intrusion Detection System) that are made to work in an SDN environment and that would be able to detect the early sign of a DoS attack to take measure to prevent. In such a case, the limit on the flow rate could be implemented only on the flow concerned and only for the time of the attack. This could be a good compromise to ensure a better security without having a heavy impact on the QoS (Quality of Service).

5.1.4. Good practices

Currently, there is a lot of things that are considered common good practices related to the security of the network that are done in a Legacy network and not necessarily implemented in SDN. We can for examples note that currently, the communication between the controller and the switches are not always encrypted which make it vulnerable to a man-in-the-middle (MITM) attack. In the same veins, there is no authentication of the switches at the controller level. This, in a nutshell, means that any malevolent can impersonate a switch and collect information or poison the network without the controller being aware of it. Implementing authorization certificates and systematically encrypting the communication of the Southbound could help diminish the importance of this threat.

5.2. Improving the Northbound security

Weve covered the Southbound and presented a few solutions to the lack of security for that part. Now, we will see what can be done on another axis of the SDN concept, the Northbound.

The source of most security failures in the Northbound is the API used. Even though a lot of different API exists, they usually are subject to the same kind of security exploits. The first and most common of those exploits was mentioned earlier in the paper and its the lack of a secured authentication on the API. This, in turn, leads to the attacker gaining access to the API and using it to implement the wrong flows on the controller which then transmit them to the whole network. If this scenario comes to pass, the attacker can perfectly create flows that would allow all kind of nefarious packet on the network or even configure the network in a way that all packet has to transit through a specific machine that would belong to the attacker. From this point, the attacker can do whatever he wants on the network.

Fortunately, a few simple solutions can help mitigate the importance of this exploit. In a nutshell, it would be to apply all the solutions existing to secure a server that are available currently. Indeed, the crux of the problem is that the API can be hacked then used for malevolent purposes. As such, to solve the problem is to secure all access to the machine hosting the API and that is no different than what is done in a legacy network. Among those methods, we can mention the use of a strong password for authentication as well as certificates and a good configuration of the ports opened on the machine to ensure that only the necessary ones are usable.

5.3. Securing the Controller

Finally, it is time to talk about the most important part to secure in an SDN network, the controller. As we have seen in an early part of this report, everything transit by the controller at some point. As such, the controller is obviously going to be the target of a lot of attacks and needs to be secure. We saw the problem of the single point failure with attacks from the Southbound. The solution to that was simply to have multiple controllers in place on the network.

Another threat to the controller is that someone takes controls of it. This is a tricky as the attempt to take control can come from the Southbound or Northbound alike, either from a connexion attempt from the network monitored or from a corrupted API. However, the solution to such problems is actually the same. To prevent fraudulent connexion to the controller it is necessary to implement a strict authentication policy with certificates to make sure that both parts of the exchange are who they claim they are. Furthermore, like before, all the method used for OS hardening (improving the security of the operating system on a machine) be it closing unnecessary ports or strong passwords are relevant for the controller.

However, there is another problem that can also an impact on the security of the whole network that involves the controller. The short explanation of this problem would be: the controller currently is only able to implement the flows he receives from the Northbound API one after another. Furthermore, the most recent one end up having the priority because the aim of SDN is to offer a dynamic monitoring of the network. If one were to only look at this way of doing things it would appear to be completely fine and perfectly aligned with the objectives of an SDN based network. In reality, this poses a huge problem on the security side of the network if this problem is not quickly addressed.

The reasoning is the following. The controllers and the API available right now are not capable of differentiating the priorities between the different protocol. This means that it is possible for a flow designed to handle one application to take precedence over an older flow meant for the security application used by the administrators. In the current situation, this is a huge problem and a potential point of entry for attackers.

Indeed, it means they could create flows that would allow a nefarious application that would block some process used by monitoring solution on the network. The difficulty in this situation is that the problem lies in the way that SDN was imagined. It is touching one of its core concept which is the dynamic gestion of the network. There is currently no real solution to this problem even if researches are done on this aspect.

6. Conclusion

As we have seen, there is still a lot of security problems in SDN and no doubt that there is still a lot of other exploits that will be found the further it is developed. in the current state of SDN, it is very easy for someone with a little bit of knowledge on SDN to find a way to attack the network. Furthermore, we have seen how big of problem an attack on the controller can be and that problem will have to be solved for SDN to fully develop into a reliable successor to legacy network. However, at the same time we saw that solutions exist to cope with most of those problems and even if these solutions are not perfect they can still mitigate the threat to the network.

In the end, we can implement every secure solution possible but we can only try to anticipate what attackers would do and it is not impossible that they still find a way to break in. As said before the concept is new, the protocols used are new and there is room for a lot of improvement still.