Global Information Assurance Certification Paper

SDN Southbound Threats

GIAC (GSEC) Gold Certification

Author: Mohamed Mahdy, mohamed.mahdy4@gmail.com
Advisor: Christopher Walker, CISSP, GCED
Accepted: 09/27/2018

Abstract

SDN (Software-Defined Networks) technologies are based on three pillars: decoupling control and forwarding planes; centralized management with a programmable network; and commodity switches. As with every new technology, the primary concern is always around security. Security concerns are on the rise due to exposing and forwarding internal communications to the network layer. For example, as a result of connecting overseas devices as a single data center or LAN, SDN infrastructure is exposed to external threats. Strategies used for SDN security are similar to legacy networks: defining the perimeters, trust areas, and stakeholders. Monitoring, including logging processes and user activity, is critical to secure the SDN components. Protection against Southbound and Northbound attacks is vital to keep the SDN deployment secured. Due to the concerns about evolving SDN threats and the different components included in their deployment, more informative penetration testing frameworks are needed to test SDN deployment security. The DELTA project (SDN evaluation framework to recognize attack cases against SDN elements and assist in identifying unknown security problems) developed by KAIST (Korea Advanced Institute of Science and Technology) students, is one such project discussed in this paper.

1. Introduction

The separation between control and forwarding planes in IP networks is inspired by the same concept used in PSTN (Public Telephony Switching Network) which serves to simplify management and provisioning. This separation was faced with fear from vendors because of the increased competition due to the standardization of control plane APIs, and network engineers because of the failure of such separation. The use of open source software for this separation was first introduced in 2008 by the Stanford University computer science department, called the Ethane project, was one of the significant practical deployments was at Google using ONIX controller in 2012.

Software-Defined Networks (SDN) can be broken down into two main channels: the Northbound interface that is responsible for the communication between the orchestrator and the control plane; and the Southbound channel which is responsible for information exchange between the control plane and the forwarding plane.

The different channels and planes introduce various threats. Using a single controller node introduces a single point of network failure. The vulnerable Northbound interface makes it easier for a malicious controller to take over the SDN infrastructure. As for the forwarding plane, the main concern includes DoS attacks that can cause congestion to the switches' interfaces or communication pipe between switches as well as controllers for checking flow rules for new flows. The data plane threats can make use of the Asynchronous communication between forwarding, and control planes to gain control over the controller or impact the SDN infrastructure. (Open Networking Foundation, 2016)

The diversity of SDN components required a framework for SDN-specific pen testing such as the one developed by The Korea Advanced Institute of Science and Technology (KAIST) student. KAIST introduced DELTA SDN pen-testing framework in the Blackhat 2016 event. SDN specific scanners provide greater visibility check both controller vulnerabilities but also vulnerabilities at multiple levels (Data plane, Management plane, controller plane, and the channels between them).

The paper mainly focuses on the Southbound interface, protocols, and the relevant threats, this interface can be used to fingerprint the whole network, cause a denial of service, and the packet headers to indirectly impact the control messages between the switches, and the controller. (Lee et al., 2017)

After this introduction, section two provides an overview of the whole SDN architecture including deployment case studies and protocols used; section three gives insights on The OpenFlow protocol, messages, headers used and how different SDN components communicate with each other; section four provides SDN threat analysis of each element in the SDN architecture, and how they can be attacked; section five focuses on SDN Southbound vulnerabilities and how they can be exploited with one of the frameworks used for SDN testing,; and section six shows some strategies to be used to secure SDN deployments.

2. SDN architecture

When SDN was first introduced, engineers and business owners started to perceive each in a way that can provide maximum benefit. Due to the flexibility of SDN, it worked well under the different implementations and for different objectives. SDN architecture provides high flexibility and agility to today's business requirements, as companies no longer are in need of vendor-specific devices at high prices.

SDN main components are the control plane, the forwarding plane, and the management plane; each plane can be equipped with a different set of protocols that suits every deployment. We'll discuss some of these deployments, and how supporting multiple protocols at each plane helps us (Open Networking Foundation, 2014).

Case 1: Customers who cannot replace their infrastructure devices with white (cheaper) boxes or prefer to have trusted vendor-managed nodes, so the SDN benefit for them is providing centralized management, this eases the task of route distribution and device management. (Open Networking Foundation, 2014)

SDN as a centralized controller deployment provides more flexibility, faster failure detection, and the ability to manipulate routes along with a Northbound interface and acts as an orchestrator that can have a programmable Infrastructure that provides better agility to business services rollout and service chaining ability.

In Case 1, each of the forwarding nodes has its own layer two protocol image, so each node within the AS (Autonomous System) can forward traffic when it receives traffic from other sources. The controller provides a central point for monitoring the whole network topology, including active and backup paths to help with route distribution (like Route Reflector in ISPs in MPLS deployment). This includes faster response to links or node failure; loop prevention and centralized admin changes to the controller node, instead of updating each node individually, Figure 1 below represents how Case1 SDN looks like.

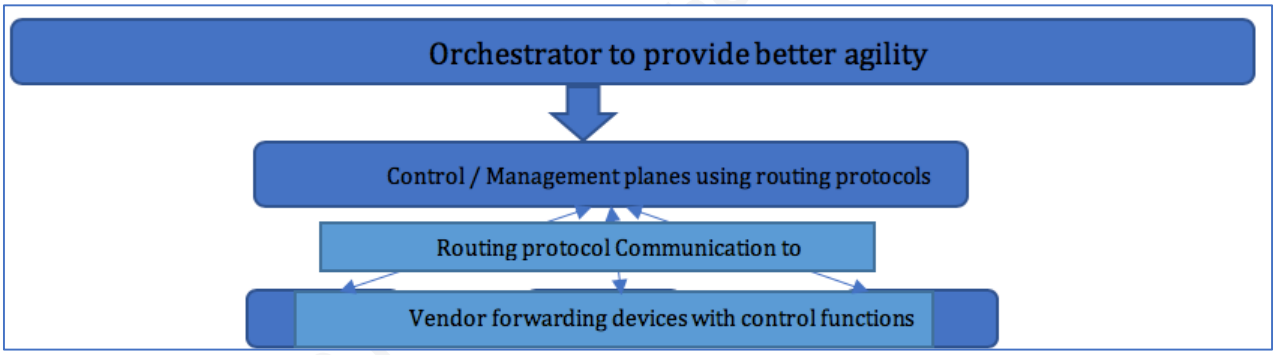


Figure 1

Case2: Other customers deploy the full SDN framework where the forwarding plane is just a controller-less "fabric", and the control plane has full control via an orchestrator that provides an interface to send instructions to the controller. (Open Networking Foundation, 2014)

In this case, as illustrated in Figure 2, the control plane uses netconf, XMPP, and OpenFlow for managing, monitoring and interacting with the forwarding nodes and the routing protocols (BGP, OSPF, ISIS) to interacts with the external nodes (external SDN controller, physical router acting as gateway) to exchange routes between different autonomous systems. The forwarding nodes rely mainly on the controller for taking the decision how to forward flows, but they have the fast-forwarding capabilities.

OpenFlow acts as one of the Southbound protocols used by the controller to communicate with the forwarding plane. At the forwarding plane, flow tables forward already known traffic; for new flows, the forwarding devices have to consult the controller to make the routing decisions.

The orchestrator interface to the controller helps engineers and service owners for better user interface access, configurations management, applying traffic policing, and updating services within a service chain to enhance security, increase business revenue or reporting services status.

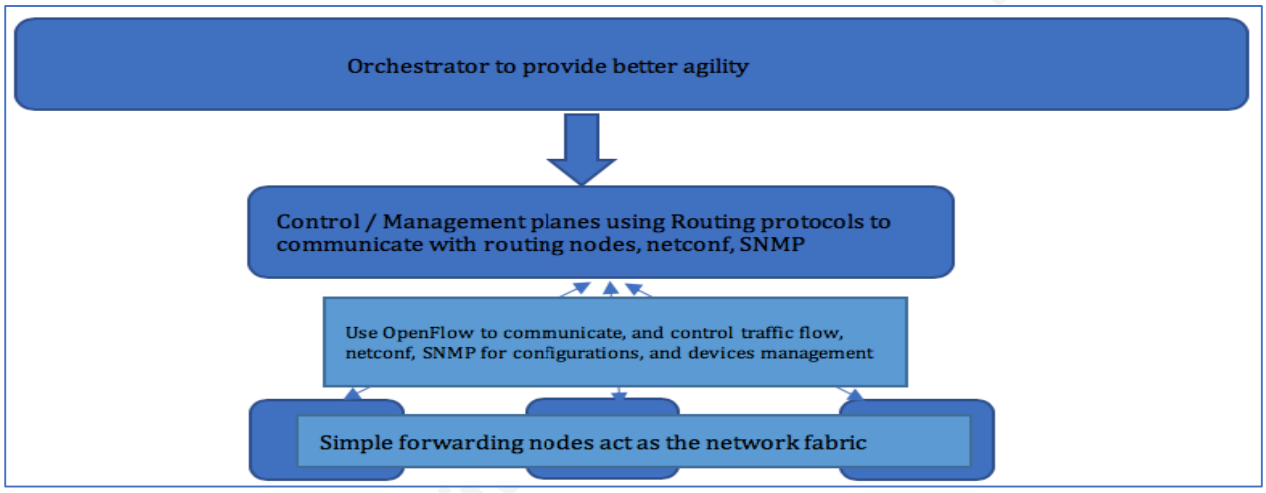


Figure 2

Case3: SDN can be viewed differently. SDN can control Networking functions through software, so every node maintains both the control, and forwarding function, but the interaction with these nodes is through a central point that communicates with other components using netconf, SNMP or the remote SSH servers to manage, configure and monitor nodes as illustrated in figure 3. (Open Networking Foundation, 2014)

Focusing on Case 2 deployments and protocols, the forwarding nodes reside in what is called the forwarding plane, controller node with the intelligence to do routing. Configuration management resides in the control plane; the management plane uses SNMP and XMPP to gather traffic statistics and monitor node performance. Also, the orchestrator Application layer provides an interface for users or other applications to interact with the controller.

The protocols and the interfaces used by the controller to communicate with the application layer are called the Northbound interface. Protocols used for communication between the controller and forwarding nodes are called Southbound interface. Northbound communication is used to retrieve info or send instructions to the controller using APIs, while the Southbound communication is used to apply configurations or update routing table via

protocol specific commands like (FLOW_MOD, PACKET_OUT) in open flow. (Open Networking Foundation, 2014)

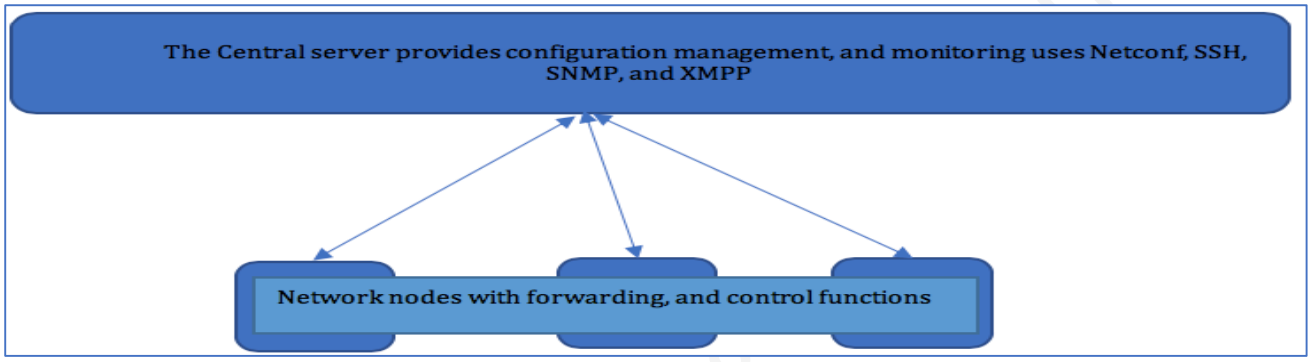


Figure 3

2.1. SDN Southbound protocols

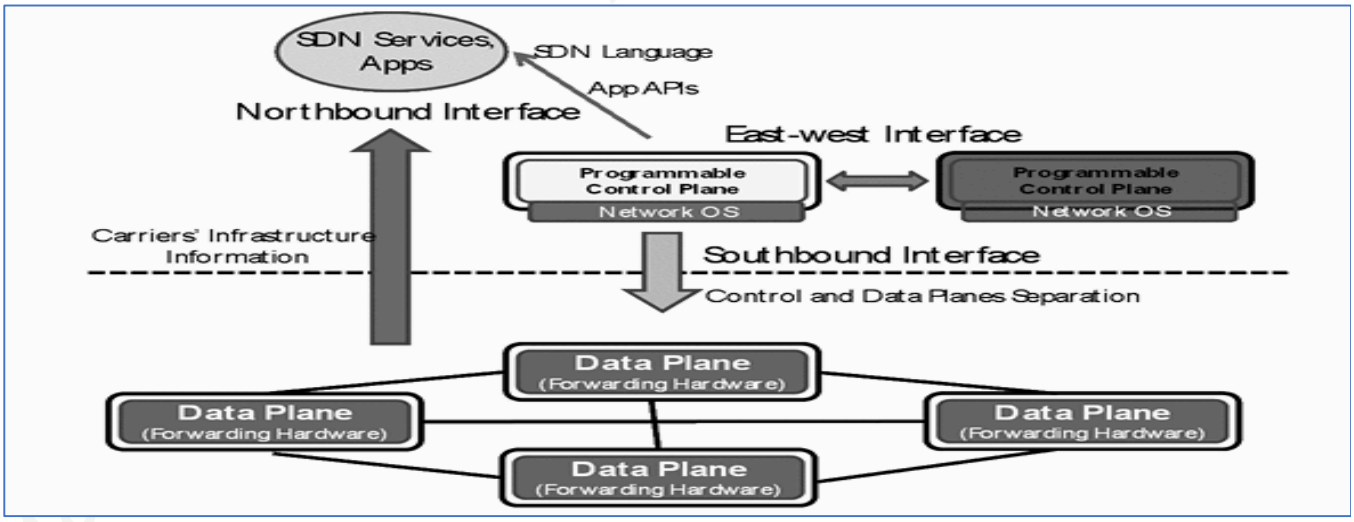


Figure 4

Figure 4 illustrates how SDN deployment supports Southbound protocols that can be used for the communication between the SDN Controller, and the forwarding plane nodes. Using Open Flow as an open standard here helps provide a better understanding of SDN concepts. (Open Networking Foundation, 2016).

Border Gateway Protocol (BGP) is used for routes exchange. The role of this protocol is critical for SDN ISP deployments. The use of this protocol can be observed in Case 1 deployments where the controller interacts with the forwarding plane nodes for routes exchange and the external routing nodes to provide a gateway for the SDN Autonomous System. BGP is being used as it provides granular routes control and interacts with external and internal AS with no issues.

Using BGP as a southbound protocol provides the nodes with a clear picture of the network which results in better failure detection, and correction, loop prevention, and easier network manageability. ("Five SDN protocols other than OpenFlow," 2014)

MPLS is used for TE-Tunnels, to build VPNs within service providers' networks, traffic resources to provide end-to-end faster failure detection, loop prevention, and better network manageability. These features appear in products like (NorthStar or Contrail introduced by Juniper, vendor-neutral solution like Tungsten Fabric).

An example to show the benefits of using MPLS, such as Southbound protocol, in Figure 5, assume that if the links between R4-R6 & R3-R5 fail, the remaining nodes to provide traffic engineering capabilities have to calculate the best path to the remote end and initiate the TE-tunnel, and VPNs over MPLS this could lead to a race between nodes over limited bandwidth links. ("Five SDN protocols other than OpenFlow," 2014)

As illustrated in Figure 5, a central node with the visibility of overall network paths checks the best tunnel per router, and pushes the configurations at once to reroute the traffic based on the available bandwidth this saves the company time and money.

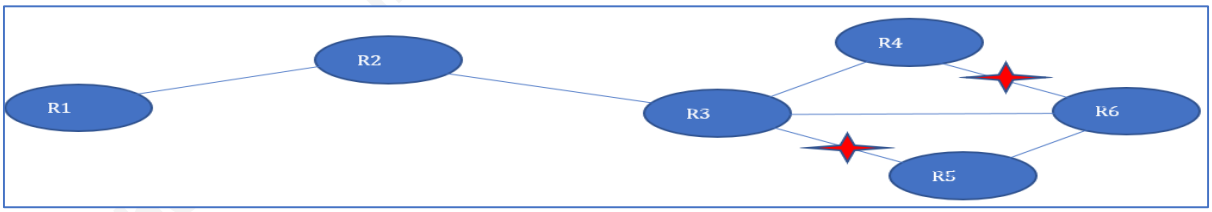


Figure 5

NetConf is used for applying configurations or extracting required output in XML format. This protocol usually runs over SSH and works well in Case 2 and Case 3 deployments as it provides a load-free method to extract devices information, and push configurations in XML format that can be efficiently processed by many scripting languages. Netconf also extracts the required XML values that we need with further automation or monitoring purposes. ("Five SDN protocols other than OpenFlow," 2014)

SNMP is critical to provide monitoring statistics that can be used for many purposes like monitoring devices' health and traffic passing through interfaces. ("Five SDN protocols other than OpenFlow," 2014)

OpenFlow This is the main protocol to be discussed in this paper. This protocol supports Case 2 as it provides full control over the forwarding plane devices, and supports Case 1 and communicates with hybrid switches that can support OpenFlow communication along with having its Layer2/Layer3 capabilities. ("Five SDN protocols other than OpenFlow," 2014)

3. OpenFlow background

OpenFlow acts as the Southbound protocol to communicate between controller, and forwarding nodes, can provide configurations management, and device monitoring through its components, and messages used.

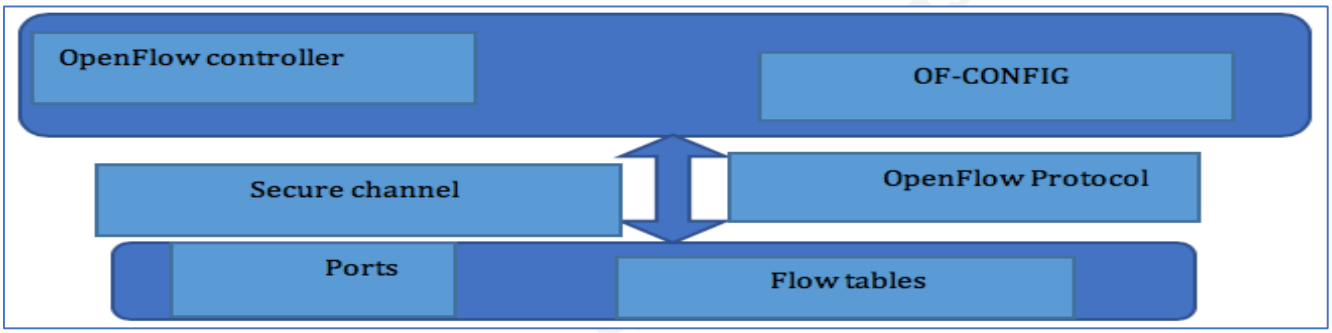


Figure 6

The OpenFlow framework as shown in Figure 6, consists of an OpenFlow controller and a configuration point at the controller plane. At the forwarding plane, there are the flow tables and entries being pushed to forward the traffic and apply required policies at the port and nested flow tables level. The communication between the controller and forwarding planes occurs using OpenFlow protocol over a secure channel. The switch consults the controller for any new or timed-out flow; then the instructions are being passed as flow entries in the flow table to be applied to the passing flows. (Marschke, Doyle, & Moyer, 2015)

As for the communication channel, while using a single device make it difficult for remote attackers to exploit that channel, in SDN deployments, this communication occurs outside the nodes using cables running through switches, and routers that are not trusted which makes these control flows easily disclosed if not well protected. OpenFlow implements TLS to secure the communication channel and makes sure messages are well encrypted preserving their integrity. (Marschke, Doyle, & Moyer, 2015)

The use of a single controller introduces a single point of failure, the solution is to use physically distributed controllers, but a single logical controller for management. Also, for the links between the controller and the forwarding nodes, multiple links should be used, called Auxiliary connections which act as backup links in case of primary link failure. They also load balance messages between switches and controller which helps to mitigate DoS due to link congestion with control messages.

OpenFlow uses different messages categories for communication between the controller, and the managed switches. These message categories are covered next. (Marschke, Doyle, & Moyer, 2015)

3.1. How OpenFlow works

OpenFlow switches as displayed in Figure 7 inspect the packets based on Layer2 and Layer3 properties like (MAC addresses, VlanID, IP address, MPLS tag) and apply the corresponding action based on the flow entries. (Marschke, Doyle, & Moyer, 2015)

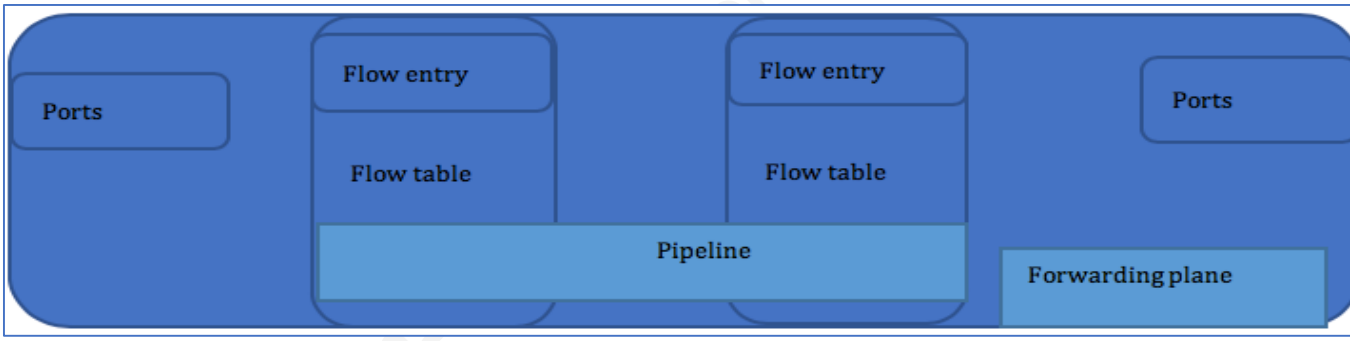


Figure 7

The traffic is being checked by the relevant flow entry; each entry checks multiple fields from the traffic as in Figure 8, assigns priorities, counts the number of flows matching this entry, and the action to be applied.

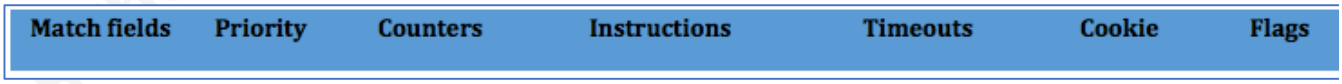
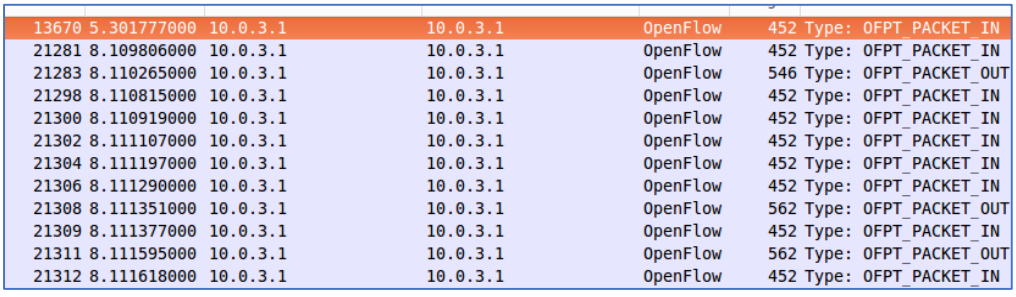


Figure 8

The flow table acts as multiple "IF" (conditional) statements, and so once a flow matches an entry from the table the instructions are applied. Due to the limited number of entries in a single table, we can use multiple tables with a pipeline connecting them, for example, traffic matching flow entry X goes to a new flow table to check other features, and so on.



13670	5.301777000	10.0.3.1	10.0.3.1	OpenFlow	452 Type: OFPT_PACKET_IN
21281	8.109806000	10.0.3.1	10.0.3.1	OpenFlow	452 Type: OFPT_PACKET_IN
21283	8.110265000	10.0.3.1	10.0.3.1	OpenFlow	546 Type: OFPT_PACKET_OUT
21298	8.110815000	10.0.3.1	10.0.3.1	OpenFlow	452 Type: OFPT_PACKET_IN
21300	8.110919000	10.0.3.1	10.0.3.1	OpenFlow	452 Type: OFPT_PACKET_IN
21302	8.111107000	10.0.3.1	10.0.3.1	OpenFlow	452 Type: OFPT_PACKET_IN
21304	8.111197000	10.0.3.1	10.0.3.1	OpenFlow	452 Type: OFPT_PACKET_IN
21306	8.111290000	10.0.3.1	10.0.3.1	OpenFlow	452 Type: OFPT_PACKET_IN
21308	8.111351000	10.0.3.1	10.0.3.1	OpenFlow	562 Type: OFPT_PACKET_OUT
21309	8.111377000	10.0.3.1	10.0.3.1	OpenFlow	452 Type: OFPT_PACKET_IN
21311	8.111595000	10.0.3.1	10.0.3.1	OpenFlow	562 Type: OFPT_PACKET_OUT
21312	8.111618000	10.0.3.1	10.0.3.1	OpenFlow	452 Type: OFPT_PACKET_IN

Figure 9

Messages communicated between the controller and the forwarding nodes are encapsulated in the OpenFlow (OF) header. Figure 9 displays an example of OpenFlow packet capture showing the messages being exchanged. (Marschke, Doyle, & Moyer, 2015)

The next section describes these messages.

3.2. OpenFlow headers, and messages

OpenFlow as a protocol is built with future development in mind; the header as shown in Figure 10 contains type, version, length (TLV) values and a transaction ID to correlate requests to responses.

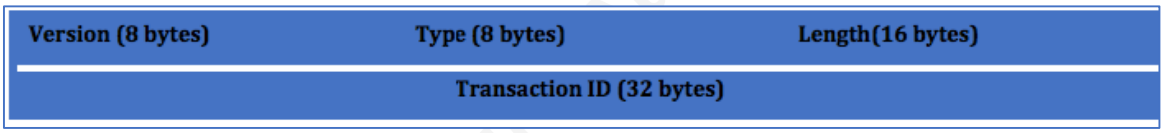


Figure 10

The messages related to OpenFlow are categorized under the three main categories controller-to-switch, asynchronous, and symmetric.

Controller-to-switch messages are used by the controller to add, modify or delete flow tables or flow tables entries. Also, the controller queries the managed switches to check the device's statistics and features and use that in determining the proper exit port for a specific flow. (Marschke, Doyle, & Moyer, 2015)

Asynchronous messages are sent by the switch to report if there's a specific flow needs a decision, port changes or error notifications.

Symmetric messages are bidirectional messages can be sent by either the switch or the controller, they act as the hello or keep-alive messages that are used for requesting an additional function.

OpenFlow messages, types, and flow processing are what enables OpenFlow to handle increased flow rates. The way flow entry is being matched is different from Forwarding Information Base (FIB), since FIB matches at the hardware level. On the other hand, the flow entry matching can be as simple as the FIB matching or advanced and apply QoS or tunneling actions.

OpenFlow matching happens at multiple levels including flow match, header match or pipeline match.

Flow match is based on flow specs like layer 2 and layer 3 addresses, port numbers, protocols, MPLS labels, and the ingress port.

Header match is similar to flow matching but only contains the packet-dependent details leaving out information about ingress port or related OF metadata.

Pipeline match occurs when the flow is forwarded between multiple internal flow tables, using the field with the input and output ports and the relevant metadata to help with forwarding the packets between different flow tables. (Marschke, Doyle, & Moyer, 2015)

The logic of OpenFlow flow processing is simple: a new packet arrives at the ingress port triggering PACKET_IN event which is sent to the controller; then the controller replies with FLOW_MOD to update the flow table with the proper flow entry; then the packet is passed whether to a new table or the egress port. (Marschke, Doyle, & Moyer, 2015)

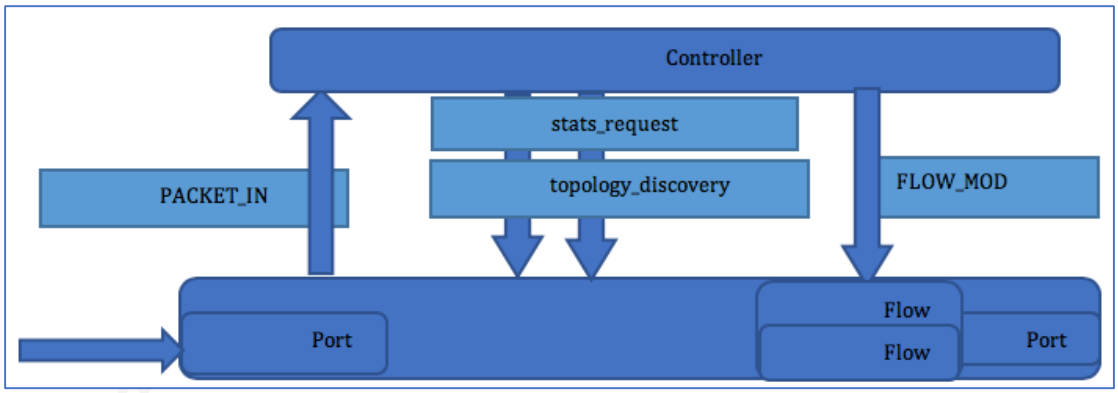


Figure 11

Packet processing flow, as seen in Figure 11, occurs when the controller receives the PACKET_IN message. It communicates with forwarding nodes to update its database through discovery and queries between the controller and the forwarding nodes, and so the controller sends stats_request_message and topology_discovery requests. Then, from the replies, it determines the proper egress port to be provided in the PACKET_OUT message delivered to the forwarding nodes. (Marschke, Doyle, & Moyer, 2015)

4. SDN Threat analysis

SDN threats are on the rise from multiple sources, and each of these sources contributes in a different way to the threats targeting the SDN deployment.

Some of these sources have a direct impact on the SDN infrastructure, others indirect. Both of these types disrupt the overlay network managed by the SDN controller.

Some of SDN threats come from vulnerable channels. These attacks are related to the communication over whether the Northbound Interface (NBI) or Southbound Interface (SBI). Some attacks are related to software, usually the result of installing untrusted packages on the controller nodes.

In this analysis, we check how hosts, switches, controllers, and orchestrators pose a threat to the SDN deployment and the vulnerabilities that are being exploited through them.

This analysis follows the definitions used by Microsoft STRIDE threat modeling process. (Open Networking Foundation, 2016)

4.1. Hosts as a source of threats

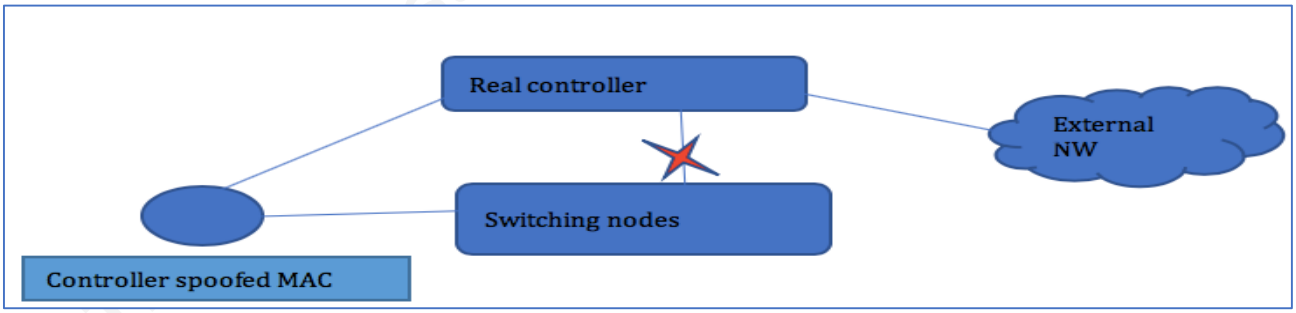


Figure 12

Users are a primary source of threats, with the current technology development users have acquired a powerful asset (whether network users or system admins) that can cause a headache to security admins. (Open Networking Foundation, 2016)

Spoofing--If no proper authentication enforced as in Figure 12, the adversary can announce himself as a controller through ARP spoofing to force traffic to pass through his device and alter or monitor them. Further, from Figure 12, an adversary can initiate direct communication with the switch claiming controller identity. (Open Networking Foundation, 2016)

Tampering--The adversary can make use of the intelligence gathered via spoofing or any other means. The adversary host machine can alter the traffic whether the statistics reported to the controller or the configurations and policies applied from the controller to the forwarding nodes. (Open Networking Foundation, 2016)

Information disclosure--Disclosure can result from the host spoofing the address or the identity of the main controller or the backup controller gaining access to the network topology, policies, and the related users' credentials. (Open Networking Foundation, 2016)

Denial of Service--These attacks are still a major threat. Threat agents can use fast MAC changing using simple tools like (macof) or start sending flows that require the switches to communicate with the controller and cause congestion on the controller-switch channel and cause DoS. (Open Networking Foundation, 2016)

```
macof -i ethx
```

Also, the user can craft packets (using Scapy) that use the same egress port at the forwarding switches that causes congestion and drop traffic at the egress interfaces.

In general, the threats generated from the hosts require protection to be close to the users' machines to limit and take faster actions toward them. Layer 2 authentication, proper mac flooding protection, proper mac or IP spoofing protection, and as usual, detection is a must we need to have visibility over user flows to make sure no abuse or malicious behavior is going on (Open Networking Foundation, 2016)

4.2. Switches as a source of threats

The forwarding nodes in the SDN deployment can cause direct or indirect harm to the SDN infrastructure. From these nodes, adversaries attack other hosts or the controller.

Switches are not entirely dummy nodes. They have firmware and flow tables that cache the controller instructions, which if not well protected can expose critical information to adversaries.

Switches can have some layer 2 protocols used for neighbor discovery and provide valuable information for the controller whether to know the topology or detect interfaces' changes (Open Networking Foundation, 2016).

Spoofing--Improper authentication between switches and controllers cause the switch to reply to API calls from adversary side and provide info to him.

Tampering--The traffic flow passing through the switch can be altered or the switch firmware itself can be changed maliciously. The integrity of the switch's operating system and the external packages used should be checked. Otherwise, an

adversary can use malformed packages to gain a backdoor to the switches and alter the flows. (Open Networking Foundation, 2016)

Information disclosure--Using spoofing techniques to gain access to private and critical data by making use of the compromised ARP table can direct the traffic to the attacker station acting as a Man-In-The-Middle. Also, malformed software packets can provide backdoor access to critical information.

Denial of Service--A compromised flow table can cause the switch to forward the traffic through the same external port, which results in network congestion. Also, a compromised flow entry can disable or ignore policies applied to limit users' traffic to comply with business requirements, and FIB processing power is also critical as with low processing power the ingress ports are easily congested. (Open Networking Foundation, 2016)

Weak flood protection can make it easy for DoS attacks sourced from users' traffic, and so supporting auxiliary communication with the controller is essential to load balance Southbound communication to prevent DoS due to Southbound channel congestion. One example of the attacks that can make use of the mutual trust between forwarding, and controller planes, and cause denial of service was discussed in CVE-2018-1000155. This exploits an improper authorization in handshake in OpenFlow v1.0 which allows the adversary to make use of DPID (Datapath Identifier) in feature_reply message where the attacker first establishes TCP connection with the controller and then initiates the handshake. (Open Networking Foundation, 2016)

4.3. Controller as a source of threats

The controller is the smart node in SDN deployment, the one that manages the forwarding nodes and communicates with external management and application planes. The controller uses multiple protocols to communicate whether via NBI or SBI.

For the controller to communicate with the application and management planes, it uses Northbound protocols like REST APIs. On the other hand, it uses BGP to communicate with external gateways; other protocols depend on the application being used. As for the communication with switching and forwarding nodes, Southbound protocols are being used to control the flow changes, query nodes status, and policies assignment.

Adding to the vulnerable channels' attacks, the node itself is being targeted, like the sync between primary and backup controllers, malicious software to be used, and update packages. (Open Networking Foundation, 2016)

Spoofing--An adversary spoofing a controller's identity can gain access to a vast amount of information. For instance, an attacker can act as a backup controller and force a sync with the primary which indicates the need for proper authentication on the different channels. As the controller communicates with external nodes, so spoofing its DNS server or its repository used for package update can still pose a threat. (Open Networking Foundation, 2016)

Tampering--Using malformed software packages, spoofed control calls, or malformed NB communication commands, the attacker can introduce changes to instructions sent by or to the controller.

Repudiation--Proving the identity of the node sending the request or the reply to a query is a must. False information provided as a reply can cause an impact on the calculated statistics.

Information disclosure--The instructions sent over controllers' communication channel is vital to the business environment as it contains business policies to be applied to users, network topology and users' credentials. Such disclosed information, if not well protected can provide access to the SDN admin machines, and from there to the whole company Network. (Open Networking Foundation, 2016)

Denial of Service--DoS is common in the controller, as the controller node can be impacted by many switches flow control messages (southbound traffic) also can be altered by the northbound traffic from the application plane. CVE-2017-1000411 shows that when multiple flows are added using REST APIs with the idle-timeout and the hard-timeout set, the expired flows stay in the controller memory and CONFIG DS (Data Store) and take up the memory despite not overloading the forwarding switches CPU nor the operational DS (Data Store). This attack can be sourced from Southbound as well when the adversary keeps sending flows to the same expired saved flow these flows are not being handled by switches due to timeout, but still consume the controller resources, and cause DoS for the controller. ("CVE-2017-1000411: OpenFlow Plugin and OpenDayLight Controller" n.d.), (Open Networking Foundation, 2016)

Elevation of Privileges--The origin of such threats can be the REST APIs communicating with the controller and can traverse specific file locations to access

restricted areas in the controller's memory to control its function. Another source can be malformed package or software update that can be used by the attacker to perform malicious activities and gain backdoor.

Controller protection requires to focus on communication via APIs from the application level, statistics reported from forwarding plane, sync with other controllers, and the admin access by the users. Each requires to network level monitoring and policing to protect the controller along with endpoint security to monitor any misbehavior at the controller level. (Open Networking Foundation, 2016)

4.4. Application plane as a source of threats

The application entry point to the SDN deployment is through the controller via the Northbound Interface (NBI). Currently, most of the NBI communication is using REST APIs, some other integrations with OSS may use shell level commands or customized scripts. (Open Networking Foundation, 2016)

The Northbound interface provides access to the SDN deployment. It is protected by security solutions that can inspect up to the application layer to detect any misbehavior that could make use of the SDN deployment.

Spoofing--An external party can spoof the identity of the application server sending the request to perform malicious activities. Also, this can happen if the application server is not well secured, and so it can be used as a proxy for the attacker's traffic towards the SDN deployment. (Open Networking Foundation, 2016)

```
curl -x http://user:password@Proxy-Ip-Here:Port required-url
```

Tampering, **--**An adversary spoofs the controller identity and acts as man-in-the-middle, so it gains the ability to alter the instructions sent by or to the application plane. If the communication is done using REST APIs with lacking proper protection, PUT method can be used to alter configurations or add malicious files that can alter other devices.

```
PUT /<malicious File> HTTP/1.1
Host: example.com
Content-type: <Format>
Content-length: 16
<p>Malicious File</p>
```

Repudiation--If no proper authentication is being used to verify the application requesting a service or calling a function, it can be repudiated. (Open Networking Foundation, 2016)

Information disclosure-- In multiple REST APIs vulnerabilities, users can use directory traversal to gather information through the exposed URL access. ("Category: Attack - OWASP," 2016), (Open Networking Foundation, 2016)

```
http://some_site.com.br/../../../../etc/shadow
```

Denial of Service--Can occur due to multiple vulnerabilities at the application level or flooding the controller with requests from the application layer. (Open Networking Foundation, 2016)

5. SDN Southbound protocols vulnerabilities

Separating control, and forwarding planes gave the network greater flexibility and agility, yet the existence of the communication path between controller and forwarding nodes (Southbound communications) introduces new attack vectors. In legacy networks, this communication used to exist within the devices, but with the introduction of SDN, this communication is external in most of the cases. These weaknesses in the Northbound and Southbound protocols that could lead to the compromise of the whole SDN infrastructure (Jero et al., 2017)

Using unencrypted channels between the Forwarding plane and controller plane could lead to eavesdropping. The attacker can make use of the unencrypted channel to monitor the messages and use that for later attacks whether tampering with the messages or gathering statistics. (Jero et al., 2017)

Weak integrity checks could lead to undetected tampering with control messages, which could lead to wrong handling of the traffic by the forwarding engine or wrong statistics sent to the controller. Using Asynchronous messages between forwarding and control planes give the attacker the chance to inject messages if no proper checking mechanisms applied.

Switch firmware still poses a large attack surface due to the vulnerabilities within the code, and the mismatch between packet header at hardware and firmware levels, this gives the attacker the ability to use covert channels methods to attack the SDN deployment. (Jero et al., 2017)

Sending control communication for new flows due to the DDoS directed to forwarding plane impacts the communication to the control plane this acts as an indirect DoS for the Southbound interface. (Jero et al., 2017)

5.1. Making use of Southbound vulnerabilities

In Figure 13, we can get an overview of the threats across the whole SDN deployment. We'll go through some techniques that were used to exploit the SDN deployment mainly via Southbound interface and forwarding plane. (Jero et al., 2017)

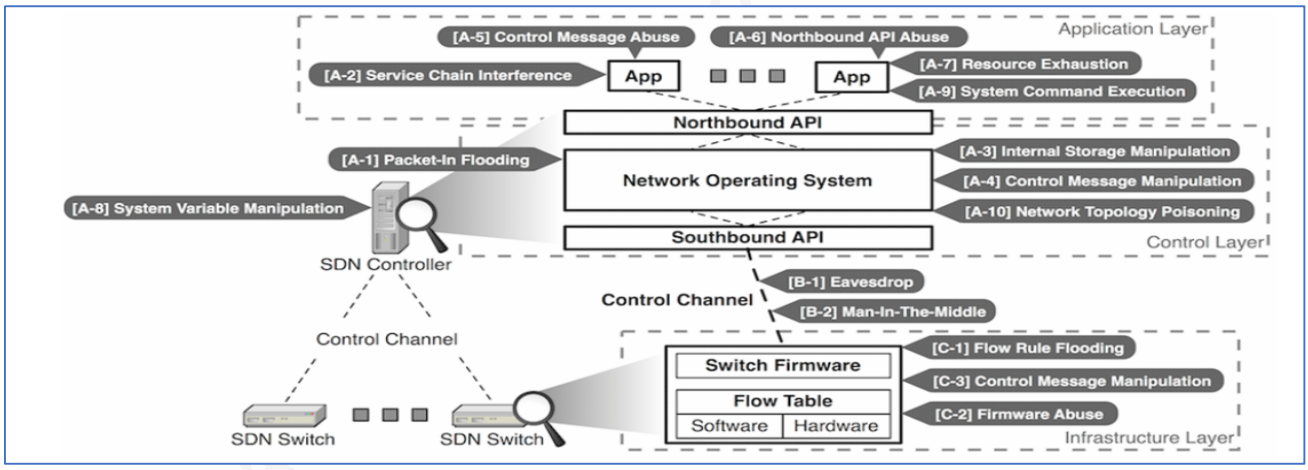


Figure 13

Further from Figure 13, delaying control messages is one of the common ways to alter SDN deployment. This can be achieved by flooding the flow table with dummy flow entries to delay more critical flows. (Jero et al., 2017)

In Figure 14, multiple different flows cause a lot of flow_add messages, and so the installed flow adds extra processing to the switch and generates more control messages ignoring other essential messages like barrier_reply, and so the controller considers the switch busy processing other messages, and no more messages to be sent to the switch. (Jero et al., 2017)

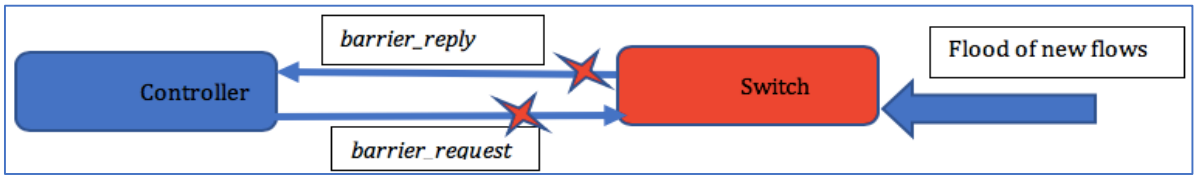


Figure 14

Disturbing the TCP OpenFlow communication between the controller, and the switch can cause the interruption of messages like packet_in and packet_out which impacts how the switch handles incoming and outgoing packets. The mentioned disruption can occur using some simple TCP known attacks. For example, initiating

duplicate TCP communication which causes the controller to drop the TCP session, changing the port in a features_reply message to be out of range at the controller side or congesting the buffers handling the control messages. TCP only needs a disruption for around 30 seconds; this can cause the session to drop, and impact the related control messages. (Jero et al., 2017)

In Figure 15, the malicious switch can provide false information to the controller like sending crafted flow_mod or port_mod messages to hide the ongoing sniffing on the network and the attacks.

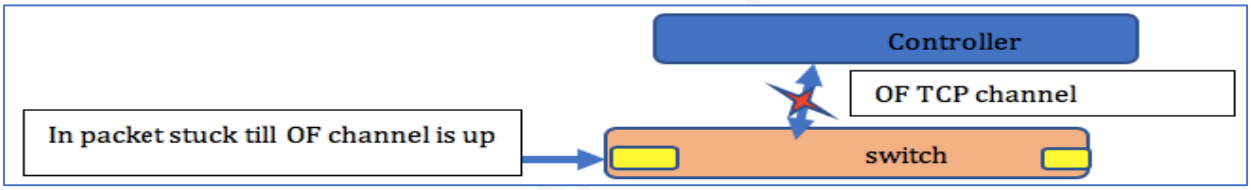


Figure 15

A compromised switch can send false IP location this impacts the SDN deployment topology and can cause high processing due to many duplicates or topology changes actions performed by the controller to respond to the received IPs locations, this can also occur by hosts using ARP spoofing (using Ettercap), and crafted topology injection messages. (Jero et al., 2017)

A crafted LLDP can be a way to provide false information about the topology setup. A mechanism was used to protect against wrong IP location which is the controller sends LLDP out of the interfaces and request port status to have a better idea adding to this monitoring the receipt of packet_in interface messages. (Jero et al., 2017)

In Figure 16, as with every protection method another abuse appeared, which is to alter the feature_list and port_status messages which makes some ports not showing in the list provided to the controller and won't be included in the calculated topology.

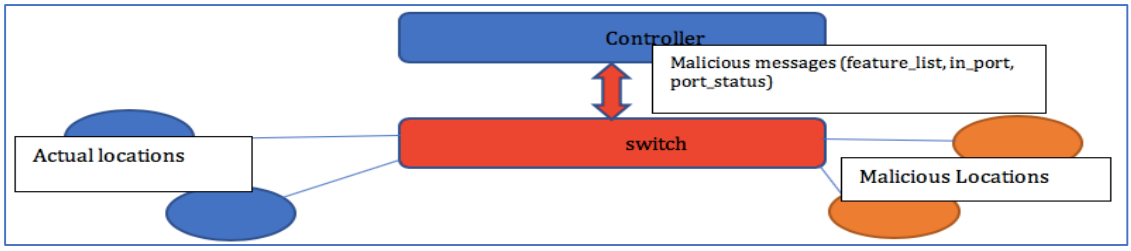


Figure 16

Another method to alter the IP location is to tamper with the broadcast message in_port. This causes the broadcast to be forwarded through the interface it received the

traffic at and so alter the learning mechanism using the source MAC and cause a change to the actual topology. (Jero et al., 2017)

Small packets can cause an issue to the SDN controller (zero-length payload) causing invalid headers to be sent the controller, and no proper learning occurs. For some controllers, zero payload packets can cause a null exception at the controller, so it's better to add restrictions as such small size messages won't have a legit use. (Jero et al., 2017)

```
alert ip any any -> any any ( sid: xxxx; rev: xx; msg: "Small size
packets"; dsize:  < 40; classtype: bad-unknown;)
```

In Figure 17, a compromised switch can make use of the idle timeouts in the flow rule entry so when the controller receives the flow_stats_reply it assumes the rule is idle, and remove it from the table causing addition delay or even impact the working flows. (Jero et al., 2017)

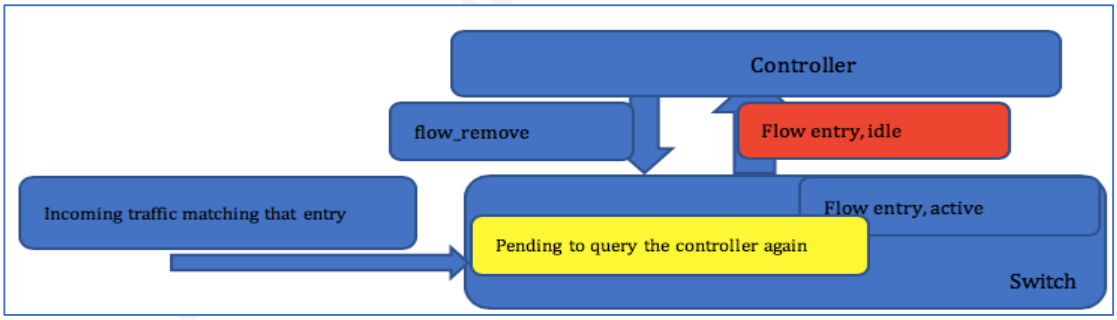


Figure 17

5.2. Frameworks to test SDN Security

There are frameworks that were developed explicitly for SDN testing, the main difference between these frameworks, and usual penetration testing frameworks is that the usual penetration testing is concerned with control plane testing, but for the SDN specific frameworks they test different levels of the deployment, the forwarding plane, the control plane, the management plane, and the channels between them.

KAIST students made significant efforts to develop penetration testing frameworks like DELTA & Poseidon which are suitable to fingerprint the SDN deployment. We'll demonstrate an example for one of the frameworks used to test SDN security which is called DELTA shown in figure 18, and this framework was developed by KAIST students, they also developed another framework called Poseidon as SDN scanner. (Lee et al., 2017)

The framework provides a set of known attacks for each plane as well as leaving the chance for fuzzing to check unknown attacks, components are illustrated in figure 18.

Agent-Manager is the control tower. It takes full control over all the agents deployed to the target SDN network. (Lee et al., 2017)

Application-Agent is a legitimate SDN application that conducts attack procedures and is controller-dependent. The known malicious functions are implemented as application-agent functions. (Lee et al., 2017)

Channel-Agent is deployed between the controller and the OpenFlow-enabled switch. The agent sniffs and modifies the unencrypted control messages. It is controller-independent. (Lee et al., 2017)

Host-Agent behaves as if it was a legitimate host participating in the target SDN network. The agent demonstrates an attack in which a host attempts to compromise the control plane. (Lee et al., 2017)

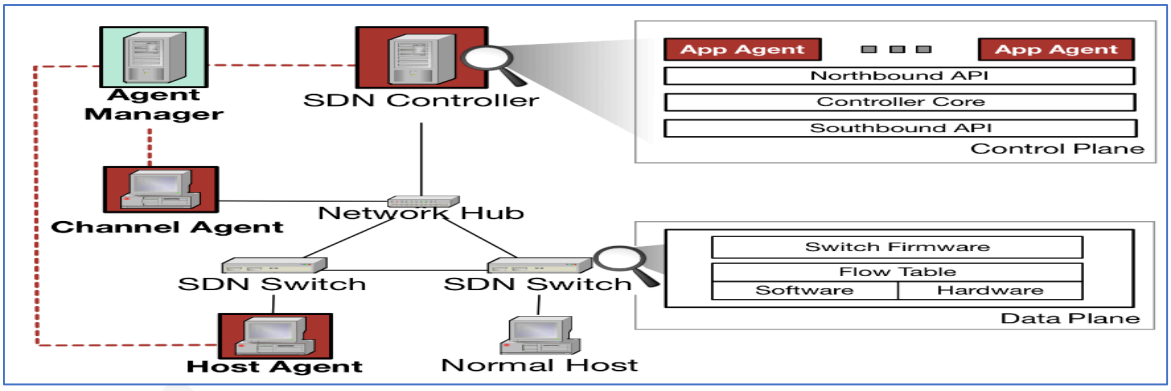


Figure 18

Below are the commands used to run the software, and how the cli interface looks like:

```
        sdntools@sdntools-VirtualBox:~/DELTA$ bin/run-delta
tools/config/trial_one.cfg
        DELTA: A Penetration Testing Framework for Software-Defined Networks
        [pP]    - Show all known attacks
        [cC]    - Show configuration info
        [kK]    - Replaying known attack(s)
        [uU]    - Finding an unknown attack
        [hH]    - Show Menu
        [qQ]    - Quit
```

The configuration file is updated as below so that we can test the Open Day Light deployment.

```
        CONTROLLER_SSH=ubuntu@DELTA_CP
        CHANNEL_SSH=ubuntu@DELTA_CH
```

```
HOST_SSH=ubuntu@DELTA_DP
TARGET_HOST=10.0.0.2
ONOS_ROOT=/home/ubuntu/onos-1.9.0
CBENCH_ROOT=/home/ubuntu/oflops/cbench/
TARGET_CONTROLLER=odl-carbon
TARGET_VERSION=6.0
OF_PORT=6633
OF_VER=1.3
MITM_NIC=eth0
CONTROLLER_IP=10.0.3.11
SWITCH_IP=10.0.3.13
DUMMY_CONT_IP=10.0.3.12
DUMMY_CONT_PORT=6633
AM_IP=10.0.3.1
AM_PORT=3366
TOPOLOGY_TYPE=VM
```

Some attacks make use of the Asynchronous nature of the communication between the controller, and the forwarding planes. For example, (OFPT_PKT_IN / PKT_OUT) messages (Disabled Table Features Request, Control Message before Hello Message). Other attacks depend on the misuse of implemented features like (TTP Port Range Violation, Table Number Violation, Malformed Version Number, Corrupted Control Message Type). Also, there're the attacks that depend on altering the configurations (Invalid OXM (OpenFlow extensible Match) – Type, length or value) along with DoS type attacks.

After running the test, it provides results similar to figure 19 about the failed, and success tests to give expectation about the SDN deployment security levels. (Lee et al., 2017)

#	Timestamp	Category	Testcase #	Name	Status	Result
1	2018-07-04 19:42:21	DATA_PLANE_OF	1.1.070	Unsupported Version Number (bad version)	COMPLETE	FAIL
2	2018-07-04 19:49:38	DATA_PLANE_OF	1.1.130	Handshake without Hello Message	COMPLETE	PASS

Figure 19

6. Securing SDN Deployment

SDN protection follows the same logic as protecting legacy networks. SDN may look different because of the separation of the forwarding, and control functions, but the security controls almost follow the same logic (Open Networking Foundation, 2016).

6.1. Define security boundaries

Securing the SDN deployment starts with defining the security boundaries of the setup. Using external vendors to handle the software threatens verifying the integrity of the delivered packages also the software security measures taken by the other vendor.

If the deployment is part of Telco cloud solution (where the Service provider provides hosting services for customers), there's a probability of receiving application plane instructions from external parties to the controller which emphasized the need to secure this channel and have full visibility on that communication to prevent malicious instructions.

At the forwarding plane, we can add distributed DDoS mitigation, and the alarming system like using Bro scripts illustrated in Figure 20, this provides a distributed security alarming tool that can alarm on a configured flows or behavior. Using Bro scripting is good for SDN framework, Figure 20 shows its ability to handle distributed nodes (Workers), and they can be managed from a central host (Manager). ("Bro Manual — Bro 2.5.4 documentation," 2016)

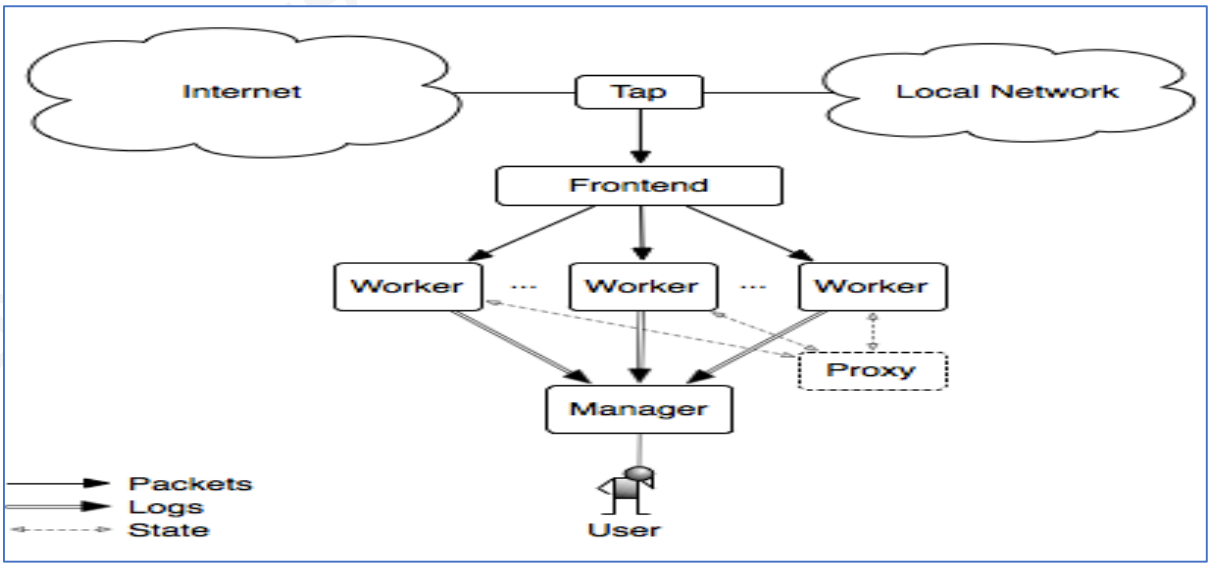


Figure 20

Having the services deployed on a virtualized platform introduce a new need for hypervisor protection, and securing the communication between the hypervisor, and virtual machines (Open Networking Foundation, 2014).

6.2. Protecting the SDN triad CIA

The main concerns about security are summarized in this CIA (Confidentiality, Integrity, and Accountability) and adding to this, Non-Repudiation. These four pillars

must be preserved in SDN setup. Encrypting the Southbound communication is critical as this interface is the most susceptible to sniffing due to usual low focus on layer 2 security, which gives the adversary a great blind spot. (Open Networking Foundation, 2016)

Ensuring TLS is enabled is critical to maintaining proper levels of Authentication, Encryption, and Integrity at both interfaces Southbound, and Northbound. (Open Networking Foundation, 2016)

Availability needs to be maintained at all levels, starting with the application plane controlling the number of APIs. Restricting the API requests processing time at the controller is essential to keep its ability to respond to forwarding plane control messages. This means that we need to have a proxy between the API senders and the controller. This helps to detect malicious requests by adding to this the firewall rules, and DoS mitigation solution to protect in case the controller API interface is exposed to public access. (Open Networking Foundation, 2016)

Accountability, and keeping logs about every user, application and process working is critical. Using API requests can make it easier for the adversary to make use of httpd related processes to access restricted areas in the memory if not well protected.

Logging and monitoring Southbound messages, connected switches, and hosts activities are critical. This makes it easier to detect suspicious activities from users or adversaries.

Having a strong accounting system that can identify the identity of the owner of each action provides greater visibility, so not only the process that's being run, but also the identity of the user who ran it.

Non-Repudiation to be addressed we need to have proper authentication, and accountability, using PKI to provide each user and the node with a key can make this an easy task. (Open Networking Foundation, 2016)

7. Conclusion

The separation between control and forwarding planes offers a more agile network that can adapt quickly to business needs. Changing the network from rigid vendor specific devices to an open standard software with standard interfaces that can be changed instantaneously to meet business requirements is far better than the need to change the hardware and different nodes configurations.

Software-based attacks can be used to manipulate network flows. The software can Impact the SDN infrastructure whether through attacks directed towards the controller as a single point of failure equipped with the intelligence and processing power to do that, or towards the forwarding and application planes.

Using OpenFlow alternatives like (BGP, XMPP, NETCONF, MPLS) exposes similar Southbound threats. These protocols can be manipulated to alter the SDN deployment whether through exchanging malicious routes or incorrect statistics.

SDN deployment is very similar to the legacy network exposures, with the internal channels being exposed to the network level. Application level attacks developed to target networks are becoming prevalent.

As SDN deployments mature with different components, using SDN-Specific scanners and pen-testing frameworks is a necessity. These must thoroughly test the SDN deployment by testing all SDN components and the channels used for the communication between these different components.

References

Open Networking Foundation. (2016). Threat Analysis for the SDN Architecture. Retrieved from: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Threat_Analysis_for_the_SDN_Architecture.pdf

Open Networking Foundation. (2014). Principles and Practices for Securing Software-Defined Networks. Retrieved from: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Principles_and_Practices_for_Securing_Software-Defined_Networks_applied_to_OFv1.3.4_V1.0.pdf

Five SDN protocols other than OpenFlow. (2014, August 28). Retrieved from https://searchsdn.techtarget.com/news/2240227714/Five-SDN-protocols-other-than-OpenFlow

Jero, S., Bu, X., Nita-Rotaru, C., Okhravi, H., Skowyra, R., & Fahmy, S. (2017). BEADS: Automated Attack Discovery in OpenFlow-Based SDN Systems. Research in Attacks, Intrusions, and Defenses, 311-333. Retrieved from: https://www.cs.purdue.edu/homes/fahmy/papers/RAID2017.pdf

Bro Manual — Bro 2.5.4 documentation. (2016). Retrieved from https://www.bro.org/sphinx/index.html

Category: Attack - OWASP. (2016, June 6). Retrieved from https://www.owasp.org/index.php/Category:Attack

Marschke, D., Doyle, J., & Moyer, P. (2015). Software Defined Networking (SDN): Anatomy of OpenFlow. s.l.: Lulu.com.

Lee, S., Yoon, C., Lee, C., Shin, S., Yegneswaran, V., & Porras, P. (2017). DELTA: A Security Assessment Framework for Software-Defined Networks. Proceedings 2017 Network and Distributed System Security Symposium.

CVE-2017-1000411: OpenFlow Plugin and OpenDayLight Controller. versions Nitrogen, Carbon, Boron, Robert Varga, Anil Vishnoi contain a flaw. (n.d.). Retrieved from https://www.cvedetails.com/cve/CVE-2017-1000411/

Appendix A

Glossary of Terms

- KAIST: Korea Advanced Institute of Science and Technology.
- SDN: Software-Defined Networks.
- DELTA: SDN evaluation framework to recognize attack cases against SDN elements and assist in identifying unknown security problems.
- ONIX: SDN controller
- FIB: Forwarding Information Base, used for forwarding frames.
- SNMP: Simple Networking Management Protocol.
- XMPP: Extensible Messaging and Presence Protocol.
- NBI: North-Bound Interface.
- SBI: South-Bound Interface.
- PKI: Public Key Infrastructure.
- API: Application Programming Interface.
- DS: Data Store.
- AS: Autonomous System.