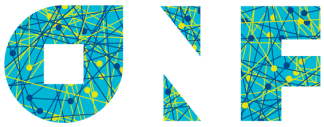

OPEN NETWORKING FOUNDATION

Threat Analysis for the SDN Architecture

Version 1.0
July 2016

TR-530

ONF Document Type: Technical Recommendations
ONF Document Name: Threat Analysis for the SDN Architecture

Disclaimer

Open Networking Foundation
2275 E. Bayshore Road, Suite 103, Palo Alto, CA 94303
www.opennetworking.org

Table of Contents

1 Introduction 5

2 Reference Model 5

3 Use Cases 7

3.1 Single-Player SDN Provider 7
3.1.1 Use Case A: Simple-level Use Case 8
3.1.2 Example of Use Case A: Cloud Service Provider 8
3.1.3 Use Case B: SDN Provider with SDN Clients and Third-Party Applications, Multi-level 9
3.1.4 Example of Use Case B: App Store 10
3.2 SDN Provider with Virtualized Network, Non-recursive 10
3.2.1 Example of Use Case C: Cloud Service Provided by Another SDN 11

4 Terminology and Acronyms 12

5 Generic Threat Modeling Approach 12

6 Threats to SDN NEs 13

6.1 Spoofing 13
6.2 Tampering 14
6.3 Repudiation 15
6.4 Information Disclosure 15
6.5 Denial of Service 15
6.6 Elevation of Privileges 16

7 Threats to SDN Controllers 16

7.1 Spoofing 16
7.2 Tampering 17
7.3 Repudiation 17
7.4 Information Disclosure 17
7.5 Denial of Service 18
7.6 Elevation of Privileges 19

8 Threats to SDN Applications 19

8.1 Spoofing 19
8.2 Tampering 20
8.3 Repudiation 20
8.4 Information Disclosure 20
8.5 Denial of Service 20
8.6 Elevation of Privileges 20

9 Conclusions 21

10 References **21**

11 Contributors **21**

List of Figures

Figure 1: OpenFlow SDN reference model, from [1] 6

Figure 2: Single-player SDN provider, from [1] 7

Figure 3: Virtual network, single-level control hierarchy, from [1] 11

Figure 4: Data Flow Diagram (DFD) for an SDN NE 13

Figure 5: DFD for SDN Controller 16

List of Tables

Table 1: Microsoft STRIDE Attack Types and Security Properties 12

1 Introduction

This document specifies the threat models and the counter-measures for the OpenFlow system, as specified in the primary SDN architecture document [1]. It will focus on the threats between different planes, i.e., the application plane, control plane, data plane, and management plane. It does not intend to specify the details of security threats and solutions to the sub-components of each plane, but these will be noted, as appropriate.

It should be noted that the threat analysis presented in this document applies to the reference SDN model from [1] in order to provide a clear picture of the relevant threats. The updated architecture document [2] introduces the concepts of resource groups and client-server contexts supporting recursion within the SDN architecture. However, the elements and interfaces of the SDN underpinning the architecture are similar in [1] and [2]. An explanation of the evolution from architecture 1.0 to 1.1 can be found in Appendix C of [2].

ONF's security principles and practices document [3] focuses on the general security principles for the SDN architecture and provides a deep security analysis with regard to the OpenFlow switch specification protocol (version 1.3.4) [4]. This current document presents an architectural threat analysis of the SDN network.

Attacks on the SDN network may result in the malfunctioning of the OpenFlow controller, a physical or logical OpenFlow switch, the management system, or SDN applications. Direct attacks on SDN applications are out of scope of the current document. However, attacks on the OpenFlow controller, switch, or management system that may result in the malfunctioning of SDN applications are within the scope of this analysis.

A brief description of potential counter-measures is also provided, with the details left to individual implementations.

2 Reference Model

The reference SDN model (see Figure 1) proposed by the Architecture Working Group [1] is composed of three planes: the application plane, the controller plane, and the data plane. There are four blocks in the reference model: the SDN-enabled application (application plane), the SDN controller (controller plane), the network element (data plane), and the operations support system. In addition, there are defined interfaces between those blocks.

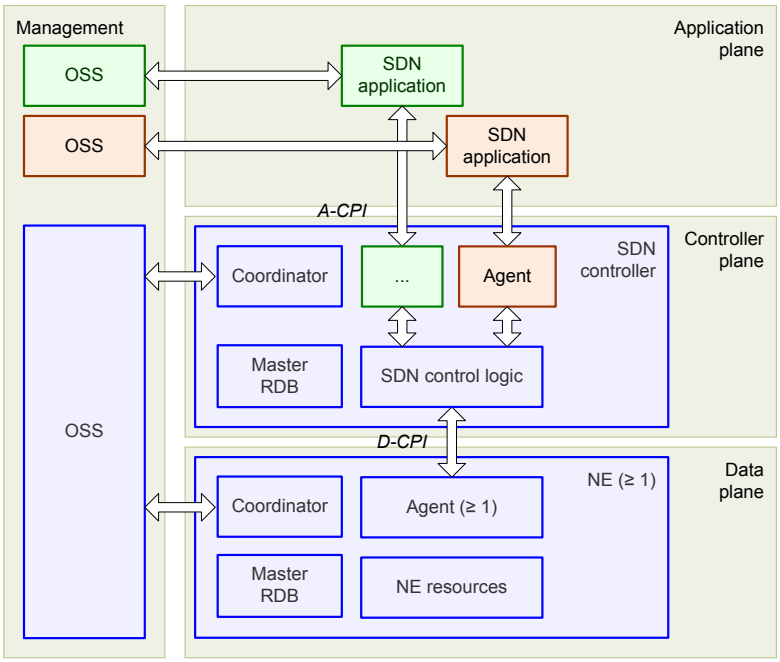

Figure 1: OpenFlow SDN reference model, from [1]

A network element (NE) is a single entity that manages a group of data plane resources. It provides a common name space used by the SDN controller to access resources that forward, manipulate, or store user data.

An SDN controller is a software entity that has exclusive control over an abstract set of data plane resources. An SDN controller may also offer an abstracted information model instance to at least one client.

An SDN-enabled application is a program that explicitly, directly, and programmatically communicates its network needs/policies/requirements/hints to the SDN controller via application interfaces (SDN network control protocols). It also consumes an abstracted view of the network for its own internal scheduling purposes.

The operations support system (OSS) is the block where all management functions are abstracted.

An SDN controller supports three functional interface types:

- A D-CPI between the data and controller planes, across which the SDN controller controls data plane resources
- An A-CPI between the application and SDN controller, across which an application receives services from the SDN controller
- A management interface, across which resources and policy may be established, as well as other more traditional management functions.

The application and the network element support management interfaces and allow establishing policies, credentials, business agreements, and so on.

3 Use Cases

There are three main use cases referred to in this document.

For the single-player SDN provider, there are two use cases.

The first is the most straightforward: all mandatory components belong to the same provider, and there is no direct tenant access to the SDN provider. This use case will be referred to as Use Case A hereafter in this document.

The second example is an SDN provider with SDN clients and third-party applications. In this use case, the SDN controller and network elements (NEs) belong to the same provider. SDN applications can either belong to the same provider, or are third-party applications that access the SDN controller through an A-CPI. This second use case is referred to as Use Case B hereafter in this document.

The third use case is an SDN provider with a virtualized network. In this scenario, the client can contract for virtual resources that span multiple NEs that are expanded on the provider controller. The client SDN controller communicates with the server SDN controller via an I-CPI. This third use case is referred to as Use Case C hereafter in this document.

3.1 Single-Player SDN Provider

Figure 2 illustrates the single-player SDN provider subnet scenario.

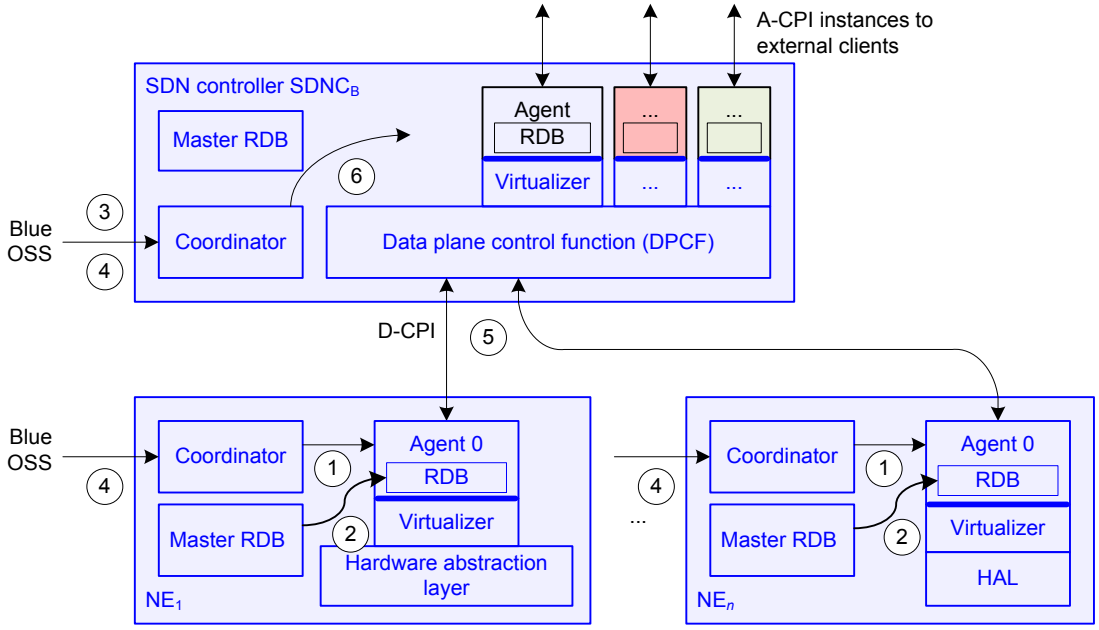

Figure 2: Single-player SDN provider, from [1]

In this scenario, the SDN controller and NEs are owned and operated by a provider, whom we will call Blue. NEs 1..n constitute the network control domain (NCD) of SDN controller $SDNC_B$ (subscript B for Blue). Everything in this scenario occurs in the Blue trust domain.

Note that for now, we do not consider the impact of internal components. In this study, we mainly focus on the four SDN elements (NEs, SDN controllers, SDN applications, and OSS) and the interactions between them.

3.1.1 Use Case A: Simple-level Use Case

In this use case, all components belong to the same provider, Blue. There is no direct tenant access to the SDN resources. OSS management can access all SDN components for management purposes.

Pre-step: All Blue components securely exchange their credentials in order to enable secure communications between different SDN components.

The use case sequence is described below:

1. Provider Blue concludes a business agreement with a tenant (out of scope for this use case) defining mutual contractual commitments of resources and service delivery.
2. Based on the tenant's needs, Blue's SDN application instructs the SDN controller to create (resp. read status/update/delete) an SDN network service for the tenant (via A-CPI). There is no direct tenant access to the resources in this use case, although the policy is enforced by the SDN controller to ensure that the tenant does not exceed the limits of the agreement. Note that the isolation between different tenants' resources is enforced by the SDN controller (at all SDN layers, including hardware and virtual resources).
3. The SDN controller interprets the requirement from the provider and generates flow rules for related SDN NEs to implement forwarding decisions (via D-CPI) and generate flow tables.
4. Provider Blue instruments its SDN controller such that the Blue OSS can collect global performance and fault statistics on its own behalf (*monitors performance*), as well as statistics associated with individual tenants (*link utilization*). These mechanisms provide information to the provider that can be used to verify SLA compliance. The statistics might also be provided to individual tenants for other purposes, if required.
5. Alarms and threshold crossing alerts are used by the provider to identify problems that require immediate action, including possibly notification to the tenant administration.
6. If Blue's business relationship with a tenant is terminated, Blue instructs the SDN controller to remove resources and any security credentials that may have been issued for that tenant. The provider must return any reserved resources to a state in which they are available for other uses. (Secure wiping of all virtual resources should be considered in this case.)

3.1.2 Example of Use Case A: Cloud Service Provider

In this scenario, the cloud service provider (Blue) has its own data centers spread worldwide. Those centers are connected by an SDN infrastructure that is operated by Blue. The communications between data centers allow realization of different tasks, such as data backup, cloud computing with remote stored data, and large user data synchronization.

When a tenant buys the cloud computing service from Blue, they sign an agreement regarding data storage volume, computing software, processing rate, and data rate. Blue employs its own SDN application to generate policies to indicate the accessible data center, bandwidth limit, routes of traffic, etc. The controller interprets the instructions received from Blue's SDN application and creates flow policies through which the related SDN switches generate the flow tables.

Each time the tenant accesses the cloud compute environment, log data should be generated and recorded, which can be checked later by the tenant.

When the stored data size exceeds the maximum assigned value, an alert message should be sent to the tenant administration. When the software of the cloud computing service crashes or when the link between switches encounters a problem, alert messages are sent to provider Blue.

If the tenant decides to terminate the service, Blue revokes the credentials and generates instructions to update the flow policies to remove the reserved resources.

3.1.3 Use Case B: SDN Provider with SDN Clients and Third-Party Applications, Multi-level

In this use case, we consider that the SDN controller and NEs belong to the same provider, Blue. We consider that SDN applications can belong to Blue, but we also assume that some SDN applications can be developed and published by third parties, such as Green. Green may also have tenants, such as Red.

In this use case, the applications have access to the Blue SDN controller through A-CPI. Blue OSS management can access all Blue SDN components and third-party applications for management purposes.

Pre-step: All Blue and Green components (Green administrator and Blue OSS/Green and Blue SDN applications/Blue SDN controllers and NEs) securely exchange their credentials in order to enable secure communications between different SDN components.

The use case sequence is described below:

1. Provider Blue authorizes Green's applications and concludes a business agreement with Green (out of scope for this use case) defining mutual contractual commitments of resources and service delivery.
2. Green's SDN applications access and instruct Blue's SDN controller (via A-CPI) to generate policies and request specific SDN resources.
3. The SDN controller instructs NEs to implement forwarding decisions (via D-CPI) in order to establish (resp. re-configure/remove) an SDN network that satisfies the application requirements.
4. Provider Blue instruments its SDN controller such that the Blue OSS can collect global performance and fault statistics on its own behalf (*monitors performance*), as well as statistics associated with individual tenants (*link utilization*). These mechanisms provide information to the provider that can be used to verify SLA compliance. The statistics might also be provided to individual tenants for other purposes, if required.

5. Alarms and threshold crossing alerts are used by the provider to identify problems that require immediate action. When necessary, the alerts are sent to Red's SDN application for tenant administration actions.
6. Whenever an application is modified (e.g. upgraded) by Green, Blue OSS should take responsibility to authorize the newer version and inform the tenants of Green about the modification.
7. If the business relationship between Green and its tenant Red is terminated, Green should revoke the permission of Red to access and configure Green's application(s). Blue OSS instructs the SDN controller to remove reserved resources.
8. If the business relationship between Green and Blue is terminated, the provider instructs the SDN controller to remove resources and removes any security credentials that may have been issued for Green. The provider must return any reserved resources to a state in which they are available for other uses. (Secure wiping of all virtual resources should be considered in this case.)

3.1.4 Example of Use Case B: App Store

SDN applications can be developed by third parties. Blue is the SDN provider in this scenario. In this example, a third-party firewall (FW) application is developed and, following authorization, published by Green in Blue's app store. (Blue's process for authorizing different SDN applications in the app store is out of scope of this document) Red, a tenant of Blue, selects and orders the applications he/she wants to use. In this case, FW and other apps are ordered.

FW is trusted by Blue. FW is configured and instructs Blue to discard traffic from some IP addresses/networks or to some IP addresses/networks for Red. Following an upgrade to FW, Blue's OSS should inform tenant Red about the upgrade, and allow Red to decide whether to transfer business rights to the latest version.

Red's administrator is able to configure the FW before running in the SDN and afterwards. Any interactions between Red administrator and FW are either indirect (contract negotiation) through the Blue OSS, or at run-time through the Blue SDN controller. In case of abnormal functionality, it is FW's responsibility to generate alert messages to the Red OSS. Blue should generate an alert message whenever the policy of FW violates the agreement between Blue and FW. Blue should also generate an alert message whenever the network elements encounter problems.

If the business relationship between Red and Green is terminated, Green should be informed to revoke Red's credential for FW upgrades. If the business relationship between Red and Blue is terminated, the Blue OSS instructs policies to the SDN network to remove all the corresponding resources reserved for Red. If Green and Blue terminate their relationship, FW should be off-shelved from the app store, and the credential between Green and Blue removed. Red must be informed about the end of functionality by FW.

3.2 SDN Provider with Virtualized Network, Non-recursive

Figure 3 demonstrates another reference model from [1] (Section 5.3). The client can contract for virtual resources that span multiple NEs, which are expanded on the server controller.

Figure 3 shows a use case with two clients, Green and Red, each with its own independent VN resource, policy, and virtualizers.

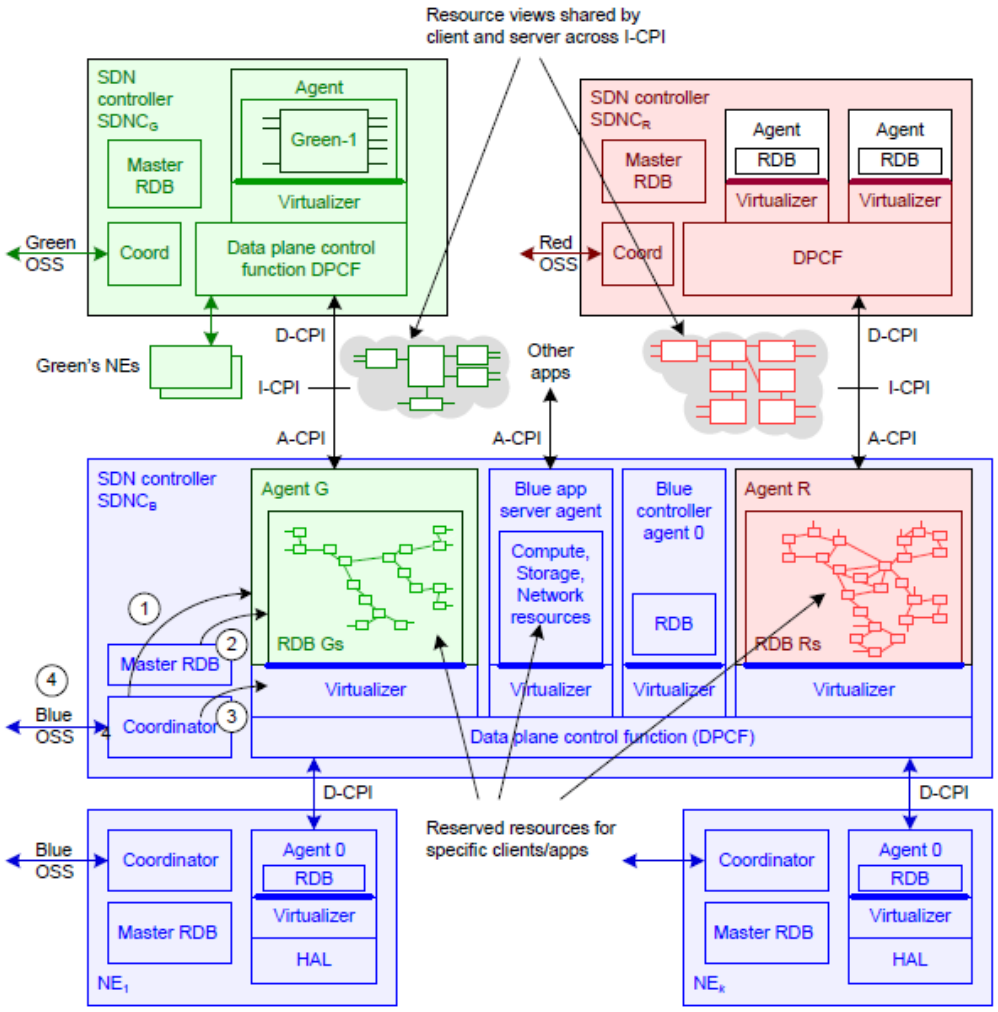

Figure 3: Virtual network, single-level control hierarchy, from [1]

3.2.1 Example of Use Case C: Cloud Service Provided by Another SDN

The cloud service provider Green has its own data centers and cloud services within its own SDN. Green can be considered as an application to other SDN providers, such as Blue.

Blue and Green reach a business agreement and establish a trust relationship. Blue then allocates the necessary resources based on its tenants' (Green and Red) SLAs. Resources are abstracted and virtualized to a pre-determined level to limit the exposure of Blue's infrastructure to Green.

Green's controller accesses the instantiated agent of Blue's controller via I-CPI to fetch relevant information and to instruct policies to orchestrate the flows from Blue to its infrastructure.

Green's OSS manages its own SDN, as does Blue's OSS. The virtualized Blue NEs are under the observation and management of Green's OSS. Thus Green's OSS can collect VNE performance and fault statistics on its own behalf, as well as statistics associated with individual tenants.

If the business relationship between Green and Blue is terminated, Blue OSS instructs policies to the SDN network to remove all the corresponding resources reserved for Green. The credential between Green and Blue is removed. Flow rules are instructed to switches to remove the related flow table items. Blue OSS generates policy to inform the controller to remove the instantiated agent for Green.

4 Terminology and Acronyms

This specification uses the terminologies and acronyms defined in [1].

5 Generic Threat Modeling Approach

The Microsoft STRIDE model is applied for each component. Each system component is considered individually—for example, the SDN controller and its interactions with other SDN components or external components. For each component, input/output and data flows are defined.

Note: The internal elements of an SDN component are not considered in this analysis. Therefore, data stores or processes are not defined in this work.

The focus is on data flows and interactions. In particular, an emphasis is placed on the SDN components' interactions with each other and with external actors. In accordance with the STRIDE methodology, different attack types are considered for each component, as shown in Table 1.

Table 1: Microsoft STRIDE Attack Types and Security Properties

Attack Type	Security Property
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-repudiation
Information disclosure	Confidentiality
Denial of service (DoS)	Availability
Elevation of privileges	Authorization

6 Threats to SDN NEs

In the threat analysis of an SDN NE, it is assumed that the NE consists of the components specified in the OF-switch [4] and OF-config [5] protocols. However, the internal implementation details of the NE are not considered.

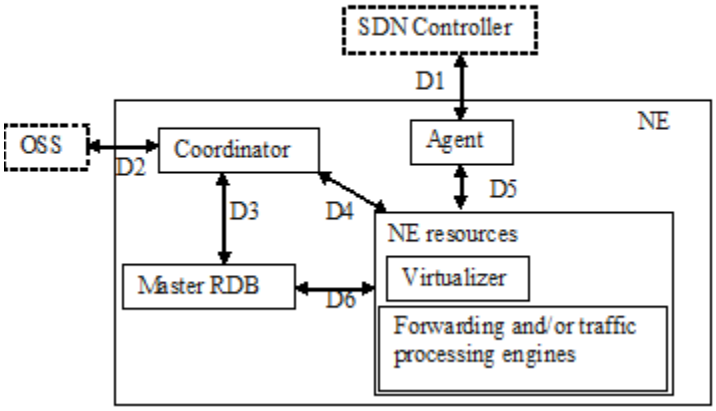

Figure 4: Data Flow Diagram (DFD) for an SDN NE

Threats to an SDN NE could come from the data plane, the control plane, and/or the management interfaces. An SDN NE could also be used to attack the SDN controller and/or the management system.

This section talks about threats to SDN NEs. These attacks apply to Use Cases A, B, and C, because there is no difference with regard to an NE and its interactions with other components in the three use cases. The only difference might be in the closed environment of Use Case A, where an attacker would have more difficulty accessing an NE than in Use Cases B and C. In this situation, the information of an NE might be somewhat disclosed to other entities through the interactions with third-party applications or the SDN controller of another domain, thus making it easier to locate an NE for the attack.

The threats to the NE from the control plane are detailed below.

6.1 Spoofing

An attacker can impersonate an OpenFlow configuration point (OFCP), which configures one or more OF capable switches via OF-config, to modify the contents of the NE's configuration database (for example, resource allocation policy to agent(s) for various controllers, the state of the NE's ports, etc.), or to modify the security-critical information (such as identifiers, reachability, and security policy and credentials, etc.) needed to establish communication with the controller(s).

By modifying some of the configuration information, an attacker can prepare for other types of attacks in the future. For example, by turning off the NE's ports, an attacker can make those

ports fail to provide a traffic forwarding service (denial of service). Alternatively, the attacker may modify the resource allocation policy in the NE to get better service than defined by its service level agreement (SLA). The attacker can also change vital information on the NE—such as the security policy and credentials of an OF controller—then spoof the controller or prepare tampering or eavesdropping attacks on the OF communication in the future. In addition, an attacker can change the NE's reachability (for example, the IP address and port number used to connect to a controller) to shut down the NE's current connection and induce the NE to reconnect to an untrusted/fake controller.

In the OpenFlow specification, mutual authentication is not mandated. This issue may be taken advantage of by attackers to perform spoofing attacks (controller process, management agent process, etc.). Then the attacker can control the OF-switch implementation to modify the NE's flow entries to redirect the traffic to its desired target for interception. When an attacker can masquerade as a controller, it can also gather some statistics information from the NE, such as its traffic processing performance, flow table capacity, etc., and then use this information for future attacks, such as DoS.

Protections: To protect the NE from spoofing attacks, strong authentication of mutual identifiers (for example, certificates signed by a trusted certificate authority (CA), or shared keys) should be mandated for communications between the control plane and the data plane, or between the data plane NEs if needed.

6.2 Tampering

If an attacker can successfully impersonate an OFCP, it can tamper with configuration data and vital data in an NE (the Master RDB in Figure 4). Additionally, when the communication messages between an OFCP and an NE are not properly protected, the messages might be tampered with by an attacker (D2); then the contents of the configuration database and/or vital information in an NE can be modified by the attacker via tampered messages. For example, the resource allocation policy could be modified for improved service; information relating to a given client could be deleted when the business agreement has not terminated; or a data port could be switched off such that hosts on that port suffer from denial of service.

The communication messages transferred between a controller and an NE are also subject to tampering when not properly protected (D1). By tampering with these messages, an attacker can perform other types of attacks on the NE. For example, an attacker can modify the policies associated with a flow to redirect the data traffic to its desired target for interception (information disclosure). When the statistics information or state changing notification messages reported by the NE are tampered with, the controller may have an incomplete or incorrect view of the NE's state, leading to incorrect traffic forwarding decisions. For example, it may assign traffic to an already congested port (a DoS attack).

Protections: To protect communication messages from being tampered with, the messages should be properly secured and a strong MAC algorithm should be employed on the message body. For example, the message body should use a digital signature digest attached to the end of the message.

6.3 Repudiation

No such issue exists here.

6.4 Information Disclosure

It is possible for an attacker to fill the flow table in order to discover its capacity (flow table storage). It is also possible for attackers to discover more information by performing side channel attacks. For instance, it may take more time for an OpenFlow switch to process a packet belonging to a new flow. Based on the discovered information, an attacker can prepare other types of attacks in the future. For example, by flooding a large volume of new flows to a low-capacity NE, the attacker can make the NE suffer from denial of service (forwarding and/or traffic processing engines).

When communications between the P-Switch and the OFCP are not encrypted, an attacker may get the chance to learn the contents of the OpenFlow packets during their transportation (D2), for example, obtaining the flow table statistical information by monitoring that communication. With the eavesdropped statistical information, the attacker can analyze the forwarding capability of the ports of the P-switch and prepare future attacks (such as a DoS attack) on the ports that have weak forwarding capability.

When communications between the P-Switch and the OFCP are not encrypted, an attacker may obtain the configuration information by monitoring the communication between the P-Switch and the OFCP. In addition, an attacker can control the OF-config implementation to release the configuration information (for example, the resources allocated to a controller) if it can successfully impersonate a legal OFCP. When those resources are released, the controller will fail to get service from the P-switch.

Protections: To mitigate this type of attack, strong encryption should be used on the communication packets.

6.5 Denial of Service

An attacker can modify the flow table entries to perform a DoS attack. For example, by letting multiple data flows from different ingress ports go out from the same egress port, it may exceed the capability of the egress port (forwarding and/or traffic processing engines). Multi-path can be used to mitigate such attacks. For instance, when a controller finds that a port is suffering packet loss through analysis of the statistical information reported by the OpenFlow switch, it should calculate another path to balance the flow out of this port, when possible.

When OpenFlow messages are transported through an encrypted channel, the processing of such messages consumes high processing power due to encryption and decryption (Agent process). As a result, an adversary may perform DoS attacks on the NE by sending a large number of bogus OpenFlow packets in a short period (forwarding and/or traffic processing engines). It is recommended that the OpenFlow channel between an OpenFlow switch and a controller should be composed of a main connection and at least one auxiliary connection. Auxiliary connections are used to improve the switch processing performance and exploit the parallelism of the switch when possible. The encryption and decryption of the communication should be processed in hardware, if possible.

Protections: Include a multi-path or auxiliary connection and resources.

6.6 Elevation of Privileges

After successfully impersonating a legal OFCP, the attacker may have an opportunity to extend its privileges (spoofing OFCP). For instance, the attacker can enhance the priority of certain queues and ensure that its data packets can be transferred faster.

Similarly, after successfully impersonating a legal controller (spoofing on controller), the attacker may have an opportunity to extend its privileges, such as by generating or modifying the flow policies in the flow tables.

Protections: Implementing mutual authentication between OFCP/OF controller and an NE, and enabling message signing.

7 Threats to SDN Controllers

In the threat analysis of an SDN controller, it is assumed that the controller consists of the components specified in the SDN architecture. However, the internal implementation details of the controller are not considered.

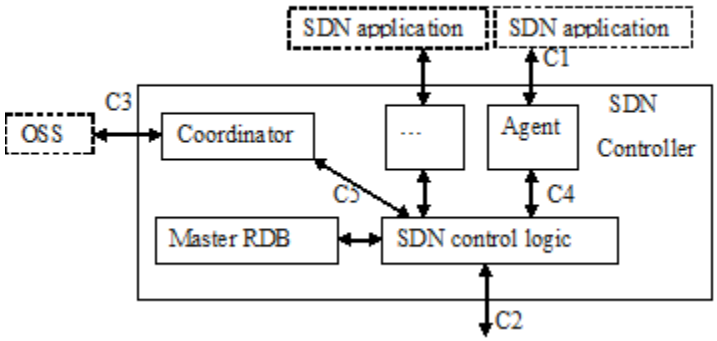

Figure 5: DFD for SDN Controller

The threats to an SDN controller are similar but with different severity levels in different use cases. In Use Case A, when it is a closed environment for all SDN components, it is more difficult to access the SDN controller because information about the controller and the communications among the controller and other components are concealed. But in Use Case B, an attacker can get information about the SDN controller through communication between a third-party application and the controller. In Use Case C, the attacker also has one more way to attack an SDN controller: through another compromised SDN controller.

7.1 Spoofing

An attacker may introduce a malicious entity into the network and send fake messages to another entity. A malicious entity can be a fake entity (for example, an NE, an application, or an OSS) that communicates with the controller (C2, C1 or C3), or it can be a fake controller (in Use Case C).

An attacker can forward packets to the SDN controller with a source address indicating that the packet is coming from a specific port or system. So the attacker can use IP spoofing to gain unauthorized access to the SDN controller.

An attacker can fake a user identity to access the SDN controller, then launch attacks on the controller, such as tampering with configuration data and management data, log information, backup flow table contents, network topology information, and other sensitive data.

> *Protections: Generally identity authentication is used to protect against spoofing. Therefore, when communicating with an SDN controller, mutual authentication should be enforced. Certificates and shared keys are common methods for identity verification.*

7.2 Tampering

Controller software or update packages may be modified by a malicious entity (for example, OSS) for attacks (C3). The attacker may intercept the supply chain to change the packages or compromise a server that is used for sending update packages.

The communication packets between the controller and any other entity (for example, an NE, an application, or an OSS) may be modified by a malicious entity in the middle of the communication (C1, C2, or C3), and any part of a packet can be modified.

Any resource data—including log information, backup flow table contents, policy, configuration data, and network topology information—can be modified in the controller. An attacker modifies the data so as to clear the attack log or prepare further attacks on OpenFlow switches. For example, an attacker can modify the controller's policies to an NE and redirect associated traffic to a specific destination for interception

> *Protections: The protection mechanism is to verify the integrity and authority of the software and update/patch packages through signatures and strong MAC (Message Authentication Code) algorithms. A recommendation here is also to sign any resource data in the controller, and enforce access control.*

7.3 Repudiation

A repudiation attack is defined as one party participating in a transaction or communication, and later claiming that the transaction or communication never took place. The controller may experience this kind of attack from applications (C1), or from upstream/downstream controllers when there are controller hierarchies.

> *Protections: The mitigation mechanism is to generate log information whenever necessary, and to enforce access control on the log information so that attackers cannot delete it.*

7.4 Information Disclosure

The risk of information disclosure includes unauthorized access to the sensitive data on the controller, such as backup flow tables, configuration data and topology data, etc.

There is also the risk of unprotected message (for example, the backup flow entries, statistics and log information) disclosure through the protocol exchange.

One critical piece of information is the cryptography key; it includes the account digital certificate private key, encryption key, encryption root key (the keys' key), etc. If the key is disclosed, the cryptography system will be destroyed. The controller usually holds the key that can verify the controller's identity. As a result, it is generally prohibited to transfer private keys via messages.

Information disclosure may also be caused by shared hardware/software resources, because different applications own data in the same controller. Application agent isolation must be enforced.

An attacker may get the OSS traffic via getting the control traffic if the NE doesn't separate OSS traffic and control traffic.

An attacker can get the opened ports, running services, and related software versions on an SDN controller by using certain scan tools. Then the attacker may use this information to attack other SDN controllers (such as a DoS attack).

For convenience, a GUI is commonly provided for ease of network management; it provides a network topology viewer, network device information, and flow table details. An attacker can easily discover that information if there is no security method on the GUI.

Protections: Enforce encryption and protect the keys. Isolate application resources in the controller.

7.5 Denial of Service

In a complex network environment such as in a MPLS carrier's core network, there are a huge number of flows. Because the network traffic is not predictive, when a packet misses the flow table lookup, it should be forwarded to the controller to cause a flow creation process (C2). The attacker may utilize the compromised switches to forward the missed packets, resulting in a DoS attack on the controller.

A cipher communication channel between the switch and the controller will increase the computing capacity shortage due to high processing during encryption and decryption.

For a centralized SDN network, a controller should be responsible for the management work of a large number of switches/applications. When the switches/applications are manipulated by attackers, they can scan the open ports/services of the controller and exchange too many concurrent messages with the controller, leading to controller overload and DoS attack.

A running application can continually request a resource (e.g., CPU or memory) of an SDN controller. If the controller does not limit an application's access to a resource, it may not be able to provide that resource to other applications.

Protections: Monitor abnormal flow table miss events with IDS, and clean/manage relevant malicious traffic. Also, enforce resource restrictions in the controller for both switches and applications.

7.6 Elevation of Privileges

The SDN controller can provide an API for third-party applications to be installed in the device, and the application can manipulate the functionality provided by the controller. If the controller cannot strictly control the privileges and the API, a malicious application may abuse the API to increase its privileges (C1).

A malicious application may make a policy that conflicts with policy from administrators or security applications to bypass the administration policy or security policy.

If the system provides some functionality to help with debugging, maintenance, service, etc., such functionality may be manipulated by a malicious user. An SDN controller provides an interface that helps a third party install, test, maintain, debug, and monitor their application. A careless design of this interface can introduce vulnerabilities in the controller. For example, a third party may get external privileges compared with the privileges approved by an AAA server, with the ability to modify the configuration directly without authorization, capture raw data for debugging that contains confidential information, etc.

An attacker may utilize the vulnerabilities of SDN controller software (e.g., software design error, software code error) to elevate the privileges of the controller.

SDN controller software can be run on a server or virtualized machine, so an attacker may utilize the vulnerabilities of the OS of the server or virtualized machine to elevate its privileges.

Protections: Strictly control the privileges of each API and the software environment.

8 Threats to SDN Applications

Threats to SDN applications in the three use cases are similar. However, in Use Case A, in the closed environment, as the SDN applications belong to the same provider as the SDN controller and NEs, it is more difficult for the attacker to get information from the inside SDN network. But in Use Cases B and C, where the applications can be third-party providers, then an attacker can get information about the controller and applications more easily. If the attacker is a third-party application itself, then the controller and applications are subject to attacks even more easily.

8.1 Spoofing

If mutual authentication is not mandated between an application and the controller, the lack of authentication may be taken advantage of by attackers to perform spoofing attacks. The information exchanged between applications and the controller includes operational statistics from the underlying physical or logical switches to applications, and intent actions from the application to the controller. One example is when an attacker masquerades as a controller to get the SLA or service-chaining of an application and use it for a future attack. An attacker can also pretend to be a controller to report incorrect statistics to applications. This can cause the application to make incorrect decisions, for example in a traffic engineering application.

If an attacker can masquerade as the controller, then it can also utilize malicious network elements under its control to serve the application. Then it will be very easy to make many kinds of attacks on the application.

Protections: Implement mutual authentication between applications and the controller.

8.2 Tampering

When strong MAC algorithms are used for message integrity protection, it can be reasonably assumed that tampering is not possible. Otherwise, an attacker may have the opportunity to tamper with messages during communications between an application and a controller. This has a severe impact on application security, because the commands from the application to the controller may be faked, and reports from the controller to the application can also be faked.

Protections: Message signing should be enforced for communications between applications and the controller.

8.3 Repudiation

The assumption is that the controller is trusted, so there will be no repudiation attack from the controller to SDN applications.

8.4 Information Disclosure

It is assumed that an SDN controller should not ask SDN applications for any especially sensitive SDN application information. However, an attacker can also get sensitive information from communications between an application and the controller. For example, if the SDN application is service function chaining, from observing the message exchange, the attacker can find out the sequence of service functions in the chain, along with other important information such as the address.

Protections: Message encryption should be enforced on communications between the application and the controller.

8.5 Denial of Service

An application might exist as an external service that is invoked by a controller—for example, a path computation engine (PCE). In a complex network environment, there might be a huge number of mouse flows. Some mouse flows may appear and then disappear after a while, then other mouse flows appear, and so on. When that happens, the controller has to invoke the PCE frequently to compute the forwarding path for these mouse flows, which can overload the PCE.

There can be other kinds of DoS attacks that are trying to consume the computing or bandwidth resources of the application.

Protections: SDN applications can mitigate a DoS attack by sending proper policies to the SDN controller with filtering functions, or by utilizing proper internal/external anti-DDoS functions.

8.6 Elevation of Privileges

No such issue exists here.

9 Conclusions

With the SDN architecture, the control plane and data plane for network elements are separated, which converts the original closed environment to an open environment. This means that the architecture is also open to attacks. For the most part, the types of attacks are not new, and there are existing mature measures to deal with them. Because SDN elements are the key for the network, it is crucial to enforce protections on them.

10 References

[1] ONF, "SDN architecture, Issue 1 June, 2014 ONF TR-502," available at: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf

[2] ONF, "SDN architecture, Issue 1.1, 2016 ONF TR-521," available at: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-521_SDN_Architecture_issue_1.1.pdf

[3] ONF, "Principles and Practices for Securing Software-Defined Networks," available at: 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

[4] ONF, "OpenFlow Switch Specification Version 1.3.4 (Protocol version 0x04)," available at: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.3.4.pdf

[5] ONF, "OF-CONFIG 1.2, OpenFlow Management and Configuration Protocol," available at: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf

11 Contributors

Ana Danping (Huawei)

Makan Pourzandi (Ericsson)

Sandra Scott-Hayward (Queen's University Belfast)

Haibin Song (Huawei)

Marcel Winandy (Huawei)

Dacheng Zhang （Huawei）

