Open SDN Controller Security

The following topics describe the security measures that Open SDN Controller implements:

Security Considerations

There are three levels of security built into Open SDN Controller: OS-level security, application-level security, and API-level security. This topic covers the security measures that are in place for each of these levels and describes any potential vulnerabilities that you should be aware of.

OS-Level Security

At the OS level, there are two main attack vectors: VM console access and SSH access. Console access is subject to VMware security measures and assumes that the client is following the guidelines VMware recommends to secure your VM console. SSH access is protected because root logins are not allowed and SSH access is disabled for all users except the sysadmin user (a user with less privileges). In addition, Open SDN Controller forces the sysadmin user to change their password after logging in for the first time and enforces password complexity requirements.

The main security vulnerability at the OS level is that the sysadmin user has sudo privileges. As a result, if the password is ever compromised, that user can get sudo root access to the system.

Application-Level Security

To address the application attack vector, Open SDN Controller redirects all HTTP traffic from port 80 to port 443, which is configured to use HTTPS to handle data. The controller also uses HTTPS to encrypt all passwords.

The main security vulnerability at the application level is that user passwords are stored in Open SDN Controller's database, meaning that the controller and user passwords reside in the same location.

API-Level Security

At the API level, Open SDN Controller uses HTTPS to handle HTTP traffic. It also minimizes password exposure in API calls by generating a token hash of the password for every call that is made. As a result, REST API calls and the password are not stored together.

Configuring LDAP

Open SDN Controller supports the use of your company's Lightweight Directory Access Protocol (LDAP) server for authentication. To enable this functionality, complete the following procedure.

Procedure

Step 1 Run the following commands to shut down the monit and controller services:

- **sudo service monit stop**
- **sudo service controller stop**

Step 2 Navigate to the following directory: /opt/cisco/controller/etc

Step 3 In a text editor, open the LDAP server configuration file (ldap.cfg).

Step 4 Locate the following settings and set the values that are specified:

- **ldap-timeout:** 3000
- **ldap-enable:** true
- **ldap-dn:** *<company-distinguished-name>*
- **ldap-ssl-port:** *<SSL-port-number>*
- **ldap-nossl-port:** *<noSSL-port-number>*
- **ldap-use-ssl:** true
- **ldap-object-group:** *<company-object-group>*
- **ldap-host:** *<LDAP-server-hostname>*

If necessary, consult your company's IT department to determine the correct values for the ldap-dn, ldap-ssl-port, ldap-nossl-port, ldap-object-group, and ldap-host settings.

Step 5 Save the changes you have made and then restart the controller.
You should now be able to log into Open SDN Controller with the username and password you use to access your company's network.

Configuring a RADIUS Server for AAA Authentication

Open SDN Controller allows you to configure a RADIUS server to implement AAA authentication. There are a number of commercial and open source RADIUS servers available for you to choose from. The following topics assume that you are configuring the FreeRadius server.

Adding a New RADIUS Server Client

The RADIUS protocol is based on UDP. Since UDP does not make use of connections, it cannot use SSL or another type of encryption based on TCP connections to handle communications. To work around this, each client that wants to use the RADIUS server for authentication must be predefined and added to the server. In FreeRadius, you accomplish this by updating the clinet.cfg file, which is located in the /etc/freeradius directory.

Note If you are using the RadiusDesk suite, the directory in which clinet.cfg resides will differ.

To add a new client, locate the following parameters in the RADIUS server's client.cfg file and define values for them:

- *client*—client's hostname
- *ipaddr*—client's IP address
- *secret*—password-like value assigned to the client

Here is what a sample client configuration looks like. The values you need to specify are italicized:

```
client cosc-ova-181 {
  ipaddr = 192.0.2.122
  secret = cosc
}
```

Configuring OSC to Use a RADIUS Server

The RADIUS configuration file, radius.cfg, is located in the /opt/cisco/controller/etc directory. It is an active file, which means that any changes made to it will automatically be rolled into OSC at runtime. As a result, you do not need to restart the controller after you edit the configuration file.

Here is an example of what the configuration file looks like:

```
radius-secret=cosc
radius-enable=true
radius-host=198.51.100.137
```

where *radius-secret* indicates the secret you defined for this client, *radius-enable* indicates whether RADIUS integration has been enabled, and *radius-host* indicates the RADIUS server's IP address.

After you have enabled RADIUS, you will be able to log into OSC with any defined RADIUS username and password combination. Note that a local OSC user is created from the RADIUS user and is assigned the User role. If you want to change the RADIUS user's role to *Admin*, you need to log into OSC as an admin user and then change that user's role from the Users page.

Setting Up TLS Support

Complete the following procedure to set up TLS support on either a Nexus 3000 Series or Catalyst 4000 Series switch.

Procedure

Step 1 Complete basic setup tasks.

a) In a directory on the controller's VM, create a subdirectory named *tls*:

- **cd** *<controller-VM-directory>*
- **bash**
- **mkdir tls**
- **cd tls**

b) Create directories for the Certification Authority (CA) certificates, private key, and CRL:

- **mkdir -p mypersonalca/certs**
- **mkdir -p mypersonalca/private**
- **mkdir -p mypersonalca/crl**
- **mkdir -p controller**
- **mkdir -p of-switch**

c) Initialize the CA database:

- **echo "01" > mypersonalca/serial**
- **touch mypersonalca/index.txt**

d) In the TLS root directory, create a file named ca.cnf (the OpenSSL configuration file) and ensure it contains the following information:

```
[ ca ]
default_ca = mypersonalca

[ mypersonalca ]
#
# WARNING: If you modify this parameter, ensure that you specify the same directory for
 the default_keyfile parameter (in the [req] section below).
# where everything resides
dir = ./mypersonalca

# where issued certificates reside
certs = $dir/certs

# where issued CRLs reside
crl_dir = $dir/crl
```

```
# database index file
database = $dir/index.txt

# default directory for new certificates
new_certs_dir = $dir/certs

#
# CA certificate
certificate = $dir/certs/ca.pem


# current serial number
serial = $dir/serial

# current CRL
crl = $dir/crl/crl.pem

# WARNING: If you modify this parameter, ensure that you specify the same directory for
 the default_keyfile parameter (in the [req] section below).
# private key
private_key = $dir/private/ca.key

# private random number file
RANDFILE = $dir/private/.rand

# extensions to add to the certificate
x509_extensions = usr_cert

# how long to certify the certificate for
default_days = 365

# how long before the next CRL
default_crl_days= 30

# which MD to use
default_md = sha1

# keep passed DN ordering
preserve = no

# section names
policy = mypolicy
x509_extensions = certificate_extensions

[ mypolicy ]
# We recommend that you do not change these values.
commonName = supplied
stateOrProvinceName = optional
countryName = optional
emailAddress = optional
organizationName = optional
organizationalUnitName = optional

[ certificate_extensions ]
# The signed certificate cannot be used as the CA.
```

```
basicConstraints = CA:false

[ req ]
# same as the private_key
default_keyfile = ./mypersonalca/private/ca.key

# specify which hash to use
default_md = sha1

# enable/disable prompts
prompt = no


# This is for CA.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer
string_mask = utf8only
basicConstraints = CA:true
distinguished_name = root_ca_distinguished_name
x509_extensions = root_ca_extensions

[ root_ca_distinguished_name ]
# update with the appropriate values for your organization.
commonName = Controller
stateOrProvinceName = Mass
countryName = US
emailAddress = root_ca_userid@cisco.com
organizationName = Cisco

[ root_ca_extensions ]
basicConstraints = CA:true
```

e) Create additional directories for the CA certificates, private key, and CRL:

- **cp ca.cnf ca_main.cnf** (ca_main.cnf acts as a backup file for ca.cnf)
- **sed s/root_ca_userid/`whoami`/ <./ca_main.cnf >./ca.cnf**
- **setenv OPENSSL ca.cnf** (for tcsh)
- **export OPENSSL="ca.cnf"** (for bash)

f) (Optional) Clean up the directories you have created before creating a new certificate in the TLS workspace:

- **cd tls**
- **rm -rf mypersonalca/index***
- **rm -rf mypersonalca/serial***
- **rm -rf mypersonalca/certs/***
- **rm -rf mypersonalca/private/***
- **rm -rf sw-cert.pem**
- **rm -rf of-switch/***
- **rm -rf controller/***

Step 2 Create the CA certificate (ca.pem) and private key (ca.key):

a) Run the following commands:

- **cd tls**
- **openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -out ./mypersonalca/certs/ca.pem -outform PEM -keyout ./mypersonalca/private/ca.key**

b) When prompted, enter the required information (such as your organization's name and your email address).

Step 3 Copy the CA certificate to the of-switch directory:
cp ./mypersonalca/certs/ca.pem ./of-switch/sw-cacert.pem

Step 4 Create the CA certificate and private key for the controller:

a) Create the controller's private key (ctl-privkey.pem) and certificate request (ctl-cert.req):

1. Run the following command:

   openssl req -nodes -newkey rsa:2048 -keyout ./controller/ctl-privkey.pem -keyform PEM -out ./controller/ctl-cert.req -outform PEM

2. When prompted, enter the required information (such as your organization's name and your email address).

b) Create the controller's CA certificate (ctl-cert.pem):
openssl ca -batch -notext -in ./controller/ctl-cert.req -out ./controller/ctl-cert.pem -config ./ca.cnf

Step 5 Verify that the certificate is valid:

a) From the controller, determine the certificate's start date and time:
openssl x509 -in ./controller/ctl-cert.pem -text | grep Not

b) From a Nexus 3000 Series switch, determine the certificate's start date and time:
sh clock

The certificate is valid when the start date and time indicated on the controller precedes the date and time indicated on the Nexus 3000 Series switch.

Step 6 Configure TLS support on your device.

- For Nexus 3000 Series switches, complete the procedure described Configuring TLS Support on a Nexus 3000 Series Switch.
- For Catalyst 4000 Series switches, complete the procedure described Configuring TLS Support on a Catalyst 4000 Series Switch.

Step 7 Configure TLS support in OSC's Openflow configuration file.
In this example, we will assume that your controller's root directory is /opt/cisco/controller/.

a) Copy ctl-cert.pem, ctl-privkey.pem, and sw-cacert.pem to the /opt/cisco/controller/configuration/certs/ directory:

- **cd tls**
- **cp controller/ctl-privkey.pem controller/ctl-cert.pem /opt/cisco/controller/configuration/certs/**
- **cp of-switch/sw-cacert.pem /opt/cisco/controller/configuration/certs/**

b) Verify that these files were copied over:

   - **cd /opt/cisco/controller/configuration/certs/**
   - **ls -al**

c) Create the TLS keystore file.

   1 Run the following commands:

   - **cd /opt/cisco/controller/configuration/certs/**
   - **cat ctl-privkey.pem ctl-cert.pem > server.pem**
   - **openssl pkcs12 -export -out server.p12 -in server.pem**

   2 Enter and then verify an export password.

   3 Run the **ls** command and verify that the following files are listed:

   - ctl-cert.pem
   - ctl-privkey.pem
   - server.p12
   - server.pem
   - sw-cacert.pem

   4 Run the following command:

   /usr/java/jdk1.7.0_75/bin/keytool -importkeystore -srckeystore server.p12 -srcstoretype pkcs12 -destkeystore ctlKeyStore -deststoretype jks

   5 Enter and then verify a destination keystore password.

   6 Enter a source keystore password.

d) Create the TLS truststore file.

   1 Run the following command:

   /usr/java/jdk1.7.0_75/bin/keytool -import -alias ca1 -file sw-cacert.pem -keystore ctlTrustStore

   2 Enter and then verify a keystore password.

   At this point, the contents of the new certificate are displayed.

   3 When prompted, enter **yes** to confirm that you want to trust this certificate.

e) Make the necessary edits to 42-openflowplugin.xml.

   1 Navigate to the /opt/cisco/controller/etc/opendaylight/karaf/ directory.

   2 In a text editor, open 42-openflowplugin.xml.

   3 Make the following changes:

   - Set the value of the transport-protocol parameter to `TLS`.
   - Uncomment any parameters that are currently commented out, like the threads parameter.

- Change any instances of `CLASSPATH` to `PATH`.
- Set the correct absolute path for both the keystore and truststore parameters.
- Set values for the keystore-password, truststore-password, and certificate-password parameters.

f) Restart the controller:

- **sudo service monit stop**
- **sudo service controller stop**
- **sudo service controller start**
- **sudo service monit start**

Configuring TLS Support on a Nexus 3000 Series Switch

Procedure

Step 1 (Optional) Open a console and run the following commands to delete the trustpoint and key that currently reside on the switch:

```
conf t
   crypto ca trustpoint myCA
   delete certificate force
   delete ca-certificate
   no rsakeypair myKey
   exit
no crypto ca trustpoint myCA
crypto key zeroize rsa myKey
```

Step 2 Set the hostname and domain name:

```
conf t
   hostname <device-name>
   ip domain-name cisco.com
```

Step 3 Create the trustpoint myCA and generate the key *myKey*.

```
crypto ca trustpoint myCA
crypto key generate rsa label myKey exportable modulus 2048
```

Step 4 Add the newly generated key to the trustpoint myCA:

```
crypto ca trustpoint myCA
   rsakeypair myKey
```

Step 5 Verify that the configuration was successful:

- **do show crypto ca trustpoints**
- **do show crypto key mypubkey rsa**
- **do show crypto ca certificates**

Step 6 Authenticate the trustpoint myCA.

a) From your TLS workspace, open the CA certificate:
cat mypersonalca/certs/ca.pem

b) Copy the certificate's text.

c) Run the following command:
crypto ca authenticate myCA

d) Paste the certificate text between the lines `-----BEGIN CERTIFICATE REQUEST-----` and `-----END CERTIFICATE REQUEST-----`.

Step 7 On the switch, generate the certificate request:

a) Run the following command:
crypto ca enroll myCA

b) When prompted, answer the questions with the responses provided in the following example:

```
Create the certificate request ..
 Create a challenge password. You will need to verbally provide this password to the CA
 Administrator
in order to revoke your certificate. For security reasons your password will not be saved
 in the
configuration. Please make a note of it.
  Password:cisco123
 The subject name in the certificate will be the name of the switch.
 Include the switch serial number in the subject name? [yes/no]:no
 Include an IP address in the subject name [yes/no]:no
 Include the Alternate Subject Name ? [yes/no]:no
 The certificate request will be displayed...
-----BEGIN CERTIFICATE REQUEST-----
MIICtDCCAZwCAQAwIDEeMBwGA1UEAxMVbng3ay0xMS1vZnAuY2lzY28uY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA82
dEVgT3nv2v2tZ6jdSq4nDRWkhj9amj8u7sbOxhvpE1jQfWT6X7EE+mvM7/zZLwgdvLS/pMAmvO2jV18M61FiXMRD2xsYDVImzDC8PH4PV/
vcV0donqAD7kl+vFQRSAL/3JlfNpVQWtAGl1Uwi2di1a1qKUC8uAd8+QGcUlXWSmJreBicoioQJW31lWzoQImQZzIsO1fznTes9ychwVsSX
1iytC8r5KFjyniQ1iYAGghTrBmtrbSEo2PmEqfIPFCX/sEDhHOFnxCPNBtYC432PU5wIUKpSsyBuKZv78/S1gLNN9Aq/coeR9dhj0TEHzLX
5QaqknZyXPOn9RRRtIdwIDAQABoE8wFwYJKoZIhvcNAQkHMQoTCGNpc2NvMTIzMDQGCSqGSIb3DQEJDjEnMCUwIwYDVR0RAQH/BBkwF4IVb
ng3ay0xMS1vZnAuY2lzY28uY29tMA0GCSqGSIb3DQEBBQUAA4IBAQALglsuqWaNX0D97719psVmXwAc2ySZuEZ+RKli2Pi/Q1U+z7f2meyX
eyR2l3k86V6x1uAeAYERdy4Dp3cFPztMvmCHN23KOAEsVcwbbrePrxySfYrhR7/XTxP3jMHlKURKZTel0rZ/Cz3YDOtbCGJ6rmUp8/pPcO
GXifPPrUMCyGXtJrjoX5SvVTUJqNorNVztazcJRWUJ55hilSThlneDVHp6NHUGe98hm4GzlQ9qVbbgtS2KrMzZbw7xSqWnDORhwS7
sLnkYf+pdX3N0mw3wbD8uvhkzJBdN8jwxGHoadEBMc3gpv1OGnxZcnYW0o77txcod99Xootykri5aK+3R
-----END CERTIFICATE REQUEST-----
```

c) Copy the text of the certificate request (which you generated in the previous step).

d) On your TLS workspace, run the following command:
vi of-switch/sw-cert.req

e) Paste the text of the certificate request into sw-cert.req between the lines `-----BEGIN CERTIFICATE REQUEST-----` and `-----END CERTIFICATE REQUEST-----`.

Step 8 Generate the switch certificate:

a) Run the following command:
openssl ca -in of-switch/sw-cert.req -out of-switch/sw-cert.pem -config ./ca.cnf

b) When prompted, enter **y** to sign the certificate.

c) When prompted, enter **y** to commit the certificate request.

Step 9 Import the CA certificate to the switch:

a) Run the following command:

cat of-switch/sw-cert.pem

b) Copy the certificate's text.

c) Run the following command:
crypto ca import myCA certificate

d) Paste the certificate text between the lines `-----BEGIN CERTIFICATE REQUEST-----` and `-----END CERTIFICATE REQUEST-----`.

e) Verify the certificate was configured:
do show crypto ca certificates

Step 10 On the switch, enter the TLS Openflow configuration:

```
openflow
  switch 1
    protocol-version negotiate
    logging flow-mod
    tls trust-point local myCA remote myCA
    probe-interval 600
    pipeline 201
    controller ipv4 10.194.132.63 port 6653 vrf management security none
    controller ipv4 10.194.132.37 port 6653 vrf management security tls
    of-port interface ethernet1/49
    of-port interface ethernet1/50
hardware profile openflow
virtual-service n3kofa
  activate
```

Configuring TLS Support on a Catalyst 4000 Series Switch

Procedure

Step 1 Clean before creating certificate and keys if already configured

```
conf t
   crypto key zeroize rsa myKey
end

conf t
   no crypto pki trustpoint myCA
end
```

Step 2 Set the hostname and domain name.

```
conf t
   hostname <device-name>
   ip domain-name cisco.com
end
```

Step 3 Set the switch's clock to a time and date that precedes the time and date set for the certificate. The command you enter should look like the following example:

```
clock set 10:53:16 6 july 2015
```

Step 4 Create a public-private keypair on the switch:

```
conf t
crypto key generate rsa general label myKey exportable
```

Step 5 When prompted, enter **2048** as the size of the key modulus for your general purpose keys.

Step 6 Verify that the key was created:
do show crypto key mypubkey rsa

Step 7 Create the trustpoint and add the private key to it:

```
conf t
   crypto pki trustpoint myCA
   revocation-check none
   rsakeypair myKey
   enrollment terminal
   subject-name CN=swA
end
```

Step 8 View the trustpoint's status:
sh crypto pki trustpoint myCA status

Step 9 Create the switch's certificate signing request (CSR):

a) Run the following command:
crypto pki enroll myCA

b) When prompted, answer the questions with the responses provided in the following example:

```
% The subject name in the certificate will include: CN=swA
% The subject name in the certificate will include: cvg-cat4k-1.cisco.com
% Include the router serial number in the subject name? [yes/no]: NO
% Include an IP address in the subject name? [no]: no
Display Certificate Request to terminal? [yes/no]: yes
Certificate Request follows:
MIICmjCCAYICAQAwNDEMMAoGA1UEAxMDc3dBMSQwIgYJKoZIhvcNAQkCFhVjdmct
Y2F0NGstMS5jaXNjby5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB
AQCivYC4X72c4DRG7uWR6x+2UIxcq6TN4722yLPAFJb/7CXhCcvBNCMObuEuBaEJ
R9NUkwdGmvU87V4Fa4rsyTf/b19fAxcKE0FtRSk11LXp346gTHMCezEzkyRqpavl
ztB5cLrQfgzBwFgFm71q+48t+vsilWBpn0iCoiMcSMZC7+zY9yrBEcZCgGwIF5og
JePy+BxigtqfOQa2gwFOTbBfg53EUgW7Aq/3TTyyCjfwvNSBZEX9IFomPWQsl0eg
bcmHDf4R953H8/Iyfo0w3Jidiy8fMnKPWcmhfK+oUMY3q0UB/DmGu46yG/I82FWM
yB6KdJy/aFwY4Vybbwye6fvXAgMBAAGgITAfBgkqhkiG9w0BCQ4xEjAQMA4GA1Ud
DwEB/wQEAwIFoDANBgkqhkiG9w0BAQUFAAOCAQEAm0kqSfaCoiCrQoXihWcnd5zl
17mWPC15MLt2tJJn9g60otcUuqRiXwxCRQG0+k5Z3VVoGwA6ObQcJ+bKXMi8dh0k
sHBpQqV8o1PFIEbwnDoc2nRVhUOCqy+Vc/FMQxJztiK9n/j0emtyTI0fl3Ae1aSh
8X1y0tAi1u/F7T9/zyCRhNQhBTIeM717Ec0y5TMbaknoUrUwRoFwqtXgpE2tvyZq
m3y990Dunjr2yC40w6HmwCEgjfvFF1YY98MwhhhY7I+WLvwNFE96/dK49Nw2xqAC
YR4V4EAZVMEOcteSUoMnp4rcl63J75ZsnxcroRKlfp02E52+DR6VyALgWm+hjQ==

---End - This line not part of the certificate request---

Redisplay enrollment request? [yes/no]: no
```

Step 10 Create the switch's certificate:

a) Copy the text of the certificate request (which you generated in the previous step).

b) Run the following commands from the tls directory:

- **mkdir of-cat4k**
- **cd of-cat4k**
- **touch sw-cert.req**

c) Paste the text of the certificate request into sw-cert.req between the lines `-----BEGIN CERTIFICATE REQUEST-----` and `-----END CERTIFICATE REQUEST-----`.

Step 11 Verify that the request was made:
openssl asn1parse -in sw-cert.req

Step 12 Sign the switch's certificate:

a) Run the following command:
openssl ca -in of-cat4k/sw-cert.req -out of-cat4k/sw-cert.pem -days 3650 -notext -config ./ca.cnf

b) When prompted, enter **y** to sign the certificate.

c) When prompted, enter **y** to commit the certificate request.

Step 13 (Optional) View the contents of the switch's certificate:
openssl x509 -in of-cat4k/sw-cert.pem -text -purpose

Step 14 Import the CA and switch certificates to the router trustpoint:

a) On the switch, run the following commands:

- **cp mypersonalca/certs/ca.pem of-cat4k/sw-cacert.pem**
- **cat sw-cacert.pem**

b) Copy the certificate's text.

c) On the router, run the following command:
crypto pki authenticate myCA

d) Paste the certificate text you copied in Step 14b between the lines `-----BEGIN CERTIFICATE REQUEST-----` and `-----END CERTIFICATE REQUEST-----`.

e) After the line `-----END CERTIFICATE REQUEST-----`, enter **`quit`**.

f) When prompted, enter **`y`** to accept the certificate.

Step 15 Import the CA certificate to the switch.

a) On the switch, run the following command:
cat sw-cert.pem

b) Copy the certificate's text.

c) On the router, run the following command:
crypto pki import myCA certificate

d) Paste the certificate text you copied in Step 15b between the lines `-----BEGIN CERTIFICATE REQUEST-----` and `-----END CERTIFICATE REQUEST-----`.

e) After the line `-----END CERTIFICATE REQUEST-----`, enter **`quit`**.

Step 16 (Optional) Verify that both the switch and CA certificates are present:
do show crypto pki cert

Step 17 On the swtich, enter the TLS OpenFlow configuration:

a) Enter the following configuration information:

```
openflow
  switch 1
   pipeline 1
   of-port interface TenGigabitEthernet1/1
   of-port interface TenGigabitEthernet1/2
   logging flow-mod
   protocol-version negotiate
   controller ipv4 <controller1-IP-address> port 6653 vrf mgmtVrf security none
   controller ipv4 <controller2-IP-address> port 6653 vrf mgmtVrf security tls
   tls trust-point local myCA remote myCA
end
```

b) Verify that the 2 controllers you just configured are listed:
sh openflow switch 1 controllers

Web Server Certificate Installation

If your company has a pre-signed certificate file, you can use that instead of the certificate file that comes with Open SDN Controller.Before you complete the following procedure, make sure that your certificate's .crt and .key files are available.

Procedure

Step 1 Copy your certificate's .crt file.

Step 2 On the machine on which Open SDN Controller is installed, navigate to the following directory:
/etc/pki/tls/certs/

Step 3 Overwrite ca.crt with your certificate's .crt file, ensuring that the filename remains ca.crt.

Step 4 Copy your certificate's .key file.

Step 5 Navigate to the following directory:
/etc/pki/tls/private/

Step 6 Overwrite ca.key with your certificate's .key file, ensuring that the filename remains ca.key.

Step 7 Restart the HTTP service by running the following command:
sudo service httpd restart

Port Usage Table

The following table lists the ports used by Open SDN Controller (in both single node and 3-node cluster setups) and their purpose. When viewing this table, note that:

- All of the ports listed below are configured to use TCP except for port 53, which uses UDP.

- Any available port can used for outgoing traffic.
- In 3 –node cluster setups, any available port can be used to transfer data between those three nodes.

Table 1: Ports Used by Open SDN Controller

Port Number	Purpose
22	Incoming and outgoing SSH traffic
53	Outbound DNS traffic
80	Incoming HTTP traffic
123	NTP connections
179	Southbound BGP connections
443	Incoming and outgoing HTTPS traffic
830	Southbound NETCONF connections
1099	Remote JMX connections
4189	Southbound PCEP connections
6633	Southbound OpenFlow connections
6653	Southbound OpenFlow connections
44444	Remote JMX connections

Supported Protocols and Services

The following table lists the protocols, TCP/IP services, and platform system services that Open SDN Controller supports.

Table 2: Protocols and Services Supported by Open SDN Controller

Protocols	
BGP-LS/PCEP	NETCONF
ICMP	OpenFlow
TCP/IP Services	
DNS	NTP

HTTPS	SSH
JMX	
Platform System Services	
cassandra	flume
collectd	httpd
controller	Java
cyanite	pathman
elasticsearch	