Table of Contents

* [Default (Wildcard) Router Certificates Lab](https://www.opentlc.com/labs/ocp_operations/02_03_Wildcard_Certificates_Lab.html#_default_wildcard_router_certificates_lab)
* [1. Prepare Environment](https://www.opentlc.com/labs/ocp_operations/02_03_Wildcard_Certificates_Lab.html#_prepare_environment)
* [2. Examine and Update Router Certificates](https://www.opentlc.com/labs/ocp_operations/02_03_Wildcard_Certificates_Lab.html#_examine_and_update_router_certificates)
  + [2.1. Examine Existing Certificate on Router](https://www.opentlc.com/labs/ocp_operations/02_03_Wildcard_Certificates_Lab.html#_examine_existing_certificate_on_router)
  + [2.2. Update Router Certificate](https://www.opentlc.com/labs/ocp_operations/02_03_Wildcard_Certificates_Lab.html#_update_router_certificate)
* [3. Configure Web Browser](https://www.opentlc.com/labs/ocp_operations/02_03_Wildcard_Certificates_Lab.html#_configure_web_browser)
  + [3.1. Copy Intermediate CA Certificate](https://www.opentlc.com/labs/ocp_operations/02_03_Wildcard_Certificates_Lab.html#_copy_intermediate_ca_certificate)
  + [3.2. Configure Your Web Browser](https://www.opentlc.com/labs/ocp_operations/02_03_Wildcard_Certificates_Lab.html#_configure_your_web_browser)
* [4. Test The Router Certificates](https://www.opentlc.com/labs/ocp_operations/02_03_Wildcard_Certificates_Lab.html#_test_the_router_certificates)
* [5. Summary](https://www.opentlc.com/labs/ocp_operations/02_03_Wildcard_Certificates_Lab.html#_summary)

Default (Wildcard) Router Certificates Lab

OpenShift’s router may be configured with a default wildcard certificate. This provides a convenient way for applications deployed to the platform to take advantage of out-of-the-box encryption without having to use their own custom certificates. This is generally a recommended practice, at least in a non-production scenario, to encourage exploration, experimentation, and rapid development.

To configure a default wildcard certificate, a certificate must be provisioned that is valid for **\*.<app domain>**, where **<app domain>**is the value of **openshift\_master\_default\_subdomain**.

Once provisioned, you must place your certificate, key, and CA certificate files on your Ansible host, and add the following line to your Ansible inventory file:

openshift\_hosted\_router\_certificate={"certfile": "/path/to/apps.c1-ocp.myorg.com.crt", "keyfile": "/path/to/apps.c1-ocp.myorg.com.key", "cafile": "/path/to/apps.c1-ocp.myorg.com.ca.crt"}

In this lab, you build on the work you completed in the previous lab. You use a wildcard certificate that was signed by the Intermediate CA. With this certificate, you access existing components without certificate warnings in your browser. You then deploy new applications that are also verified with the same Master CA certificate via the trust chain established with the Intermediate CA certificate.

1. Prepare Environment

1. Log in to your OpenShift Bastion host and switch to the **root** user.
2. Set a shell variable for the router’s address on the infrastructure node—the port number is **443**:

ROUTER=$(perl -ne 's/^(infranode1\..\*?\.internal).\*$/$1/ and print and last' /etc/ansible/hosts):443; echo ${ROUTER}

1. Set an environment variable with your environment’s GUID.

export GUID=$(hostname | cut -f2 -d.); echo ${GUID}

We’ve created some simple scripts to create the necessary certificates and keys for your environment. Use them throughout the GPTE DevOps certificate labs.

1. If they don’t already exist from Lab 1 (**ls /root/certs/ca**), generate the new Intermediate CA keys and special server keys on your Bastion host.
2. cd /root/
3. git clone https://github.com/newgoliath/certs/
4. cd certs
5. ./ca\_create.sh

./create\_env\_certs.sh -g ${GUID}

2. Examine and Update Router Certificates

2.1. Examine Existing Certificate on Router

1. Run the OpenSSL tool on the bastion host:

echo QUIT | openssl s\_client -connect ${ROUTER} 2>&1 | more

**Sample Output**

depth=1 /CN=openshift-service-serving-signer@1504798915

verify error:num=19:self signed certificate in certificate chain

verify return:0

CONNECTED(00000003)

---

Certificate chain

0 s:/CN=router.default.svc

i:/CN=openshift-service-serving-signer@1504798915

1 s:/CN=openshift-service-serving-signer@1504798915

i:/CN=openshift-service-serving-signer@1504798915

---

Server certificate

-----BEGIN CERTIFICATE-----

<OMITTTED>

-----END CERTIFICATE-----

subject=/CN=router.default.svc

issuer=/CN=openshift-service-serving-signer@1504798915

---

No client certificate CA names sent

---

SSL handshake has read 2644 bytes and written 456 bytes

---

New, TLSv1/SSLv3, Cipher is DHE-RSA-AES128-SHA

Server public key is 2048 bit

Secure Renegotiation IS supported

Compression: NONE

Expansion: NONE

SSL-Session:

Protocol : TLSv1

Cipher : DHE-RSA-AES128-SHA

Session-ID: 97BB9AF23D6CA1D65F1E3DF3E14EE4E570491BE905A62B20DFFDE491C5F04BB1

Session-ID-ctx:

Master-Key: 8A995996A88B78B0C00CD17113A9F684F0479DD09712DE62911E7485742011ADB8178B77FF84BF0FD167241D93355E25

Key-Arg : None

Start Time: 1505331134

Timeout : 300 (sec)

Verify return code: 19 (self signed certificate in certificate chain)

---

HTTP/1.0 408 Request Time-out

Cache-Control: no-cache

Connection: close

Content-Type: text/html

<html><body><h1>408 Request Time-out</h1>

Your browser didn't send a complete request in time.

</body></html>

closed

1. The new wildcard certificate, key, and Intermediate CA certificate can be found in your **/root/certs/ca/intermediate/newcerts/${GUID}/** directory.
2. To configure Ansible to deploy the new router certificates, add the following lines to the **/etc/ansible/hosts** Ansible inventory file, in the **[OSEv3:vars]** section:

|  |  |
| --- | --- |
|  | Due to a bug in the playbooks, you **must replace ${guid} with your GUID** in the following variable. |

openshift\_hosted\_router\_certificate={"certfile": "/root/certs/ca/intermediate/newcerts/${GUID}/wildcard.apps.${GUID}.example.opentlc.com.cert.pem", "keyfile": "/root/certs/ca/intermediate/newcerts/${GUID}/wildcard.apps.${GUID}.example.opentlc.com.key.pem", "cafile": "/root/certs/ca/intermediate/certs/intermediate.cert.pem"}

|  |  |
| --- | --- |
|  | In this example, you use the same CA file for the router as you did for the Platform CA so that you might not need to add a second certificate to the clients, including your web browser. Usually there is a separate CA certificate for these applications. This is especially the case if the applications being served at this URL are external URLs. |
|  | A best practice is to generate fully qualified domain names in your certificates for each of your applications' domain names. The wildcard is a convenience to aid quick development of multiple applications. |

2.2. Update Router Certificate

1. Run the Ansible Playbooks:

ansible-playbook -i /etc/ansible/hosts -f 20 -v /usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/redeploy-router-certificates.yml

1. Examine the router for the new certificate and proper common name (CN):

echo QUIT | openssl s\_client -connect ${ROUTER} 2>&1 | more

* + Expect the output to show the certificate signed by the Intermediate CA certificate:

**Sample Output**

depth=0 /C=US/ST=North Carolina/L=Raleigh/O=Red Hat, Inc./OU=GPTE DevOps/CN=\*.apps.GUID.example.opentlc.com/emailAddress=gpte-devops-automation@redhat.com

verify error:num=20:unable to get local issuer certificate

verify return:1

depth=0 /C=US/ST=North Carolina/L=Raleigh/O=Red Hat, Inc./OU=GPTE DevOps/CN=\*.apps.GUID.example.opentlc.com/emailAddress=gpte-devops-automation@redhat.com

verify error:num=27:certificate not trusted

verify return:1

depth=0 /C=US/ST=North Carolina/L=Raleigh/O=Red Hat, Inc./OU=GPTE DevOps/CN=\*.apps.GUID.example.opentlc.com/emailAddress=gpte-devops-automation@redhat.com

verify error:num=21:unable to verify the first certificate

verify return:1

CONNECTED(00000003)

---

Certificate chain

0 s:/C=US/ST=North Carolina/L=Raleigh/O=Red Hat, Inc./OU=GPTE DevOps/CN=\*.apps.GUID.example.opentlc.com/emailAddress=gpte-devops-automation@redhat.com

i:/C=US/ST=North Carolina/O=Red Hat, Inc./OU=GPTE DevOps/CN=Red Hat OpenTLC Classroom Intermediate CA/emailAddress=gpte-devops-automation@redhat.com

1 s:/C=US/ST=North Carolina/L=Raleigh/O=Red Hat, Inc./OU=GPTE DevOps/CN=Red Hat OpenTLC Classroom Root CA/emailAddress=gpte-devops-automation@redhat.com

i:/C=US/ST=North Carolina/L=Raleigh/O=Red Hat, Inc./OU=GPTE DevOps/CN=Red Hat OpenTLC Classroom Root CA/emailAddress=gpte-devops-automation@redhat.com

---

Server certificate

-----BEGIN CERTIFICATE-----

<OMITTED>

-----END CERTIFICATE-----

subject=/C=US/ST=North Carolina/L=Raleigh/O=Red Hat, Inc./OU=GPTE DevOps/CN=\*.apps.GUID.example.opentlc.com/emailAddress=gpte-devops-automation@redhat.com

issuer=/C=US/ST=North Carolina/O=Red Hat, Inc./OU=GPTE DevOps/CN=Red Hat OpenTLC Classroom Intermediate CA/emailAddress=gpte-devops-automation@redhat.com

---

No client certificate CA names sent

---

SSL handshake has read 4290 bytes and written 456 bytes

---

New, TLSv1/SSLv3, Cipher is DHE-RSA-AES128-SHA

Server public key is 2048 bit

Secure Renegotiation IS supported

Compression: NONE

Expansion: NONE

SSL-Session:

Protocol : TLSv1

Cipher : DHE-RSA-AES128-SHA

Session-ID: 4DA438AD0EF3D52DBF9DA84C90AB9CE7D774D93CCFEBDDD9148C9CD9F0BB6F86

Session-ID-ctx:

Master-Key: 13428471C716FB511A744B2C83E59B0D10EB27BFE616E5F3EEE99F730080B6E765598FEAFF1231839B2FD0517AAD8EF1

Key-Arg : None

Start Time: 1505334039

Timeout : 300 (sec)

Verify return code: 21 (unable to verify the first certificate)

---

DONE

* + This example uses the **GUID**. Your deployment will have your own **${GUID}**. The new certificate has a **subject** value that includes the wildcard CN (beginning with **\***), and an **issuer** that is the Intermediate CA.

3. Configure Web Browser

3.1. Copy Intermediate CA Certificate

1. On the bastion host, copy the Intermediate certificate to your user’s home directory in order to **scp** it to your laptop.
2. sudo -i
3. cp /root/certs/ca/intermediate/certs/intermediate.cert.pem /home/<OpenTLC username>/

chown <OpenTLC username> /home/<OpenTLC username>/intermediate.cert.pem

1. On your laptop, download the following file via scp from your bastion to your laptop using **scp**:

scp -i <Your OpenTLC private key> <OpenTLC username>@bastion.${GUID}.example.opentlc.com:intermediate.cert.pem .

* + This is the Intermediate CA certificate that signed most of OpenShift’s server certificates. Note the location it was downloaded.

3.2. Configure Your Web Browser

Select the following directions that reflect your client platform:

1. Firefox 57 on Fedora
2. Chromium on Fedora
3. Firefox 57 on Mac
4. Chrome on Mac

3.2.1. Configure Firefox Web Browser CA on Linux Fedora 27

1. In Firefox, navigate to **about:preferences#privacy**.
2. Scroll down the page, and select **View Certificates → Authorities**:
3. Click **Import** and select the Intermediate CA certificate file from your download location.
4. Check the **Trust this CA to identify websites** box and click **OK.**
5. Click **OK** again to close the certificates box.
6. The Intermediate CA Certificate is now properly imported into Firefox 57+ on Fedora. You may now proceed to Step 3.3 which tests the Intermediate CA Certificate.

3.2.2. Configure Chromium Web Browser CA on Linux Fedora 27

In this section, you add the Intermediate CA certificate to your Linux operating system’s Chromium browser’s certificate management component.

1. In Chromium, navigate to **chrome://settings/?search=Manage%20certificates** and click **Manage Certificates**.
2. Select **Authorities → Import**.
3. From the file selection dialog box, select the Intermediate CA certificate file and click **Open**.
4. Tick the checkbox that indicates **Trust thie certificate for identifying websites**, and click **OK**.
5. You should see the Red Hat OpenTLC Classroom Intermediate CA" in the list of trusted authorities as depicted here:
6. The Intermediate CA Certificate is now properly imported into Chromium on Fedora. You may now proceed to Step 3.3 which tests the Intermediate CA Certificate.

3.2.3. Configure Firefox Web Browser CA on Mac

1. In Firefox, navigate to **about:preferences#privacy**.
2. Scroll down the page, and select **View Certificates → Authorities**:
3. Click **Import** and select the Intermediate CA certificate file from your download location.
4. Check the **Trust this CA to identify websites** box and click **OK.**
5. Click **OK** again to close the certificates box.
6. The Intermediate CA Certificate is now properly imported into Firefox 57+ on Mac. You may now proceed to Step 3.3 which tests the Intermediate CA Certificate.

3.2.4. Configure Chrome Web Browser CA on Mac

In this section, you add the Intermediate CA certificate to your Mac operating system’s certificate management application.

1. In Chrome, navigate to **chrome://settings/?search=Manage%20certificates** and click **Manage Certificates**.
   * Expect your operating system’s certificate management box to appear.

|  |  |
| --- | --- |
|  | The instructions that follow are for the macOS operating system. |

1. Import the Intermediate CA file into the System keychain. Click the **System** keychain, and then click the padlock above **Click to unlock the System keychain**:
2. Enter your password and click **Modify Keychain**.
3. At the bottom of the box, click the section "Certificates" and then the **+** to add the certificate.
4. From the file selection dialog box, select the Intermediate CA certificate file and click **Open**.
5. If prompted for your macOS password, enter it to import the certificate.
6. Select the name of the new Intermediate CA from the list and type **command+I** to invoke the **Get Info** dialog for this certificate.
7. Click the triangle to expand the **Trust** section.
8. Select the list box next to **Secure Sockets Layer (SSL)** and set it to **Always Trust**, and close the window.
9. Enter your password again if prompted.
10. Your Keychain Access application should include the following highlighted line:
11. Click the **lock** icon to lock the keychain. Enter your password if required.
12. Close the **Keychain Access** window.
13. The Intermediate CA Certificate is now properly imported into Chrome on Mac. You may now proceed to Step 3.3 which tests the Intermediate CA Certificate.

4. Test The Router Certificates

1. Navigate to the following URLs:
   * **https://hawkular-metrics.apps.${GUID}.example.opentlc.com/hawkular/metrics**
   * **https://kibana.apps.${GUID}.example.opentlc.com**
2. Note that you are not prompted to accept the use of self-signed certificates.

5. Summary

Default wildcard certificates are useful for allowing developers and development managers to create applications that are accessible securely. They’re very useful in proof of concept environments. They are also somewhat useful in production environments, as long as strict checks are in place to keep application proliferation well documented and controlled. They are less desireable in public Internet facing applications - as they can quite easily lead to applications getting leaded to the wrong populations. For public facing applications, it’s recommended to use per-application certificates. Look for labs teaching the use of automated certificate generation for applications in the near future.

Build Version: be32934b41da1e34c27e21e79bdf57d30c732a3e : Last updated 2018-01-19 05:59:04 EST