THALES

OpenShift Container Platform Image Signing

INTEGRATION GUIDE

THALES LUNA HSM

Document Information

Document Part Number	007-001278-001
Revision	A
Release Date	3 June 2021

Trademarks, Copyrights, and Third-Party Software

CONTENTS

Overview

Red Hat® OpenShift® is a CNCF certified Kubernetes platform and distribution solution. Red Hat OpenShift offers a consistent hybrid cloud foundation for building and scaling containerized applications. Luna HSMs enable you to store the cryptographic keys used for signing the container images deployed in OpenShift. With container image signing you can validate the integrity of container image and where it came from. Following are some of the benefits of using Luna HSMs along with OpenShift container-based applications:

- Secure generation, storage, and protection of cryptographic keys on FIPS 140-2 level 3 validated hardware.
- Full life cycle management of keys.
- HSM audit trail.

Certified Platforms

This integration is certified for Luna HSM on the following platforms:

HSM Type	GPG Server OS	OpenShift Container Platform
Luna HSM	RHEL 7	4.6

Luna HSM: Luna HSM appliances are purposefully designed to provide a balance of security, high performance, and usability that makes them an ideal choice for enterprise, financial, and government organizations. Luna HSMs physically and logically secure cryptographic keys and accelerate cryptographic processing. The Luna HSM on premise offerings include the Luna Network HSM, Luna PCIe HSM, and Luna USB HSMs. Luna HSMs are also available for access as an offering from cloud service providers such as IBM cloud HSM, and AWS cloud HSM classic.

Prerequisites

Before you proceed with the integration, complete the following tasks:

- Configure Luna HSM
- Set up OpenShift Container Platform
- Set up GPG server

Configure Luna HSM

To configure Luna HSM:

1. Ensure that the HSM is set up, initialized, provisioned, and ready for deployment.
2. Create a partition on the HSM for use by OpenShift Container Platform.
3. Create and exchange certificate between the Luna Network HSM and client system. Register client and assign partition to create an NTLS connection. Initialize Crypto Officer and Crypto User roles for the registered partition.
4. Verify that the partition is successfully registered and configured. The command to see the registered partition is:

   ```
   # /usr/safenet/lunaclient/bin/lunacm
   ```

```
lunacm (64-bit) v10.2.0-111. Copyright (c) 2020 SafeNet. All rights reserved.

        Available HSMs:
        Slot Id ->              0
        Label ->                ocp
        Serial Number ->        1238696044950
        Model ->                LunaSA 7.4.0
        Firmware Version ->     7.4.0
        Configuration ->        Luna User Partition With SO (PW) Key Export
                                With Cloning Mode
        Slot Description ->     Net Token Slot
        FM HW Status ->         Non-FM
        Current Slot Id: 0
```

5. For PED-authenticated HSM, enable partition policies 22 and 23 to allow activation and auto-activation.

> **NOTE:** Refer to Luna HSM documentation for detailed steps for creating NTLS connection, initializing the partitions, and managing various user roles.

Set up OpenShift Container Platform

Refer to OpenShift Documentation for installing and running the OpenShift Container Platform.

Set up GPG server

Connect to any Red Hat Enterprise Linux (RHEL 7) host as a user with administrative privileges and perform the following tasks:

- Install GPG-dependent packages
- Install Pinentry package
- Install GPG package
- Install PKCD11-Helper and GnuPG-PKCS11 SCD
- Install Skopeo

Install GPG-dependent packages

Install the following GPG-dependent packages from https://www.gnupg.org/download/index.html:

- npth
- libgpg-error
- libgcrypt
- libksba
- libassuan

> **NOTE:** The GPG-dependent packages that you install should be compatible with gpg version 2.0.22.

Install Pinentry package

To authenticate partition access for GPG, you can use the **salogin** utility, which gets automatically installed along with the Thales Luna client software. However, if you do not want to use this utility, you can install the Pinentry package available at https://www.gnupg.org/download/index.html.

> **NOTE:** The Pinentry package that you install should be compatible with gpg version 2.0.22.

Install GPG package

After building and installing the above packages, you need to install the GPG version 2.0.22 package available at https://www.gnupg.org/download/index.html

Install PKCS11-Helper and GnuPG-PKCS11 SCD

After installing GPG, you need to install pkcs11-helper and gnupg-pkcs11-scd and libraries.

- pkcs11-helper (https://github.com/OpenSC/pkcs11-helper/releases)
- gnupg-pkcs11-scd version 0.9.2 (https://github.com/alonbl/gnupg-pkcs11-scd/releases/)

> **NOTE:** While building the gnupg-pkcs11-scd daemon, the development packages associated with these libraries are subsequently used at runtime. To keep the new library versions separate from the versions that are already installed, run the export LD_LIBRARY_PATH=/usr/local/lib command.

Install Skopeo

Install Skopeo rpm package: `yum install skopeo.`

Integrating GnuPG with Luna HSM

To Integrate GnuPG with Luna HSM, you need to complete the following tasks:

> Access Luna HSM

> Configure the gnupg-pcs11-scd.conf file

> Generate Keys and Certificates

> Configure GPG to use the PKCS#11 Smart Card Daemon

Access Luna HSM

You can use either of the following methods to access Luna HSM on GPG:

> Use salogin utility

> Use Pinentry

Use salogin utility

The persistent session allows the GPG to access the HSM object without prompting the password every time.

To open the persistent session using salogin utility, perform the following steps:

1. Add the following text in the /etc/Chrystoki.conf file:

```
Misc = {
   AppIdMajor=1;
   AppIdMinor=1;
}
```

2. Run the following command to open the authenticated persistent session to access the HSM object:

```
# ./salogin -o -s 0 -i 1:1 -p <partition_password>
```

   Where `-s` represents the slot_id and `-i` represents the AppId set in the Chrystoki.conf file.

Use Pinentry

To use Pinentry, you need to add the following text in the /root/.gnupg/gpg-agent.conf file:

```
pinentry-program /usr/local/bin/pinentry
```

> **NOTE:** If gpg-agent.conf file isn't available, you need to create it at /root/.gnupg/ directory.

Configure the gnupg-pcs11-scd.conf file

> **NOTE:** Skip this step if you are using the salogin utility.

To configure the gnupg-pkcs11-scd.conf file, add or modify the following lines in the /root/.gnupg/gnupg-pkcs11-scd.conf file:

```
provider-p11-allow-protected-auth
```

```
provider-p1-cert-private
provider-p1-private-mask 0
```

> **NOTE**: If the gnupg-pkcs11-scd.conf file isn't available, you need create this file and copy all the contents to it from /usr/local/etc/gnupg-pkcs11-scd.conf.example or /usr/local/share/doc/gnupg-pkcs11-scd/gnupg-pkcs11-scd.conf.

Generate Keys and Certificates

After creating the NTLS connection with HSM, follow these steps to generate the RSA key pair on HSM. GPG uses several asymmetric key pairs as part of keychain configuration. These are signing, encryption, and authentication key pairs. It is possible to use the same key pair for all three functions. To generate keys and certificates:

1. Generate the RSA key pair on Luna HSM using the CMU utility provided with Luna Client in /usr/safenet/lunaclient/bin directory. Provide the partition password when prompted.

```
# ./cmu generatekeypair -modulusBits=2048 -publicExponent=65537 -
labelPublic=GPG-Sign-Pub –labelPrivate=GPG-Sign-Priv -id=11111101 -sign=T -
verify=T -encrypt=T -decrypt=T

Please enter password for token in slot 0 : ********

Select RSA Mechanism Type -

[1] PKCS [2] FIPS 186-3 Only Primes [3] FIPS 186-3 Auxiliary Primes : 1
```

   Select RSA Mechanism Type as [1] PKCS

2. List the contents generated on HSM partition and note down the handle of public/private key. Provide the partition password when prompted.

```
# ./cmu list

Please enter password for token in slot 0 : ********

handle=34       label=GPG-Sign-Pub

handle=35       label=GPG-Sign-Priv
```

3. Generate the self-signed certificate from the generated public/private key. Provide the partition password and certificate attributes when prompted.

```
# ./cmu selfsigncertificate -publichandle=34 -privatehandle=35 -
startDate=20200225 -endDate=20251025 -serialNumber=0133337A -
keyusage=digitalsignature,keyencipherment -label=GPG-Sign

Please enter password for token in slot 0 : ********

Enter Subject 2-letter Country Code (C) : IN

Enter Subject State or Province Name (S) : UPST

Enter Subject Locality Name (L) : NOIDA

Enter Subject Organization Name (O) : GEMALTO

Enter Subject Organization Unit Name (OU) : IDPS

Enter Subject Common Name (CN) : GPG-Signing

Enter EMAIL Address (E) :
```

> **NOTE:** Use self-signed certificate only for test purposes. In the production environment, create the certificate request and get it signed by a Trusted Certificate Authority.

4. If you want to use different encryption and authentication keys/certificates, then repeat the above steps.

> **NOTE:** Ensure that the **id** and **label** for every key/certificate is different.

Configure GPG to use the PKCS#11 smart card daemon

Perform the following steps to configure the gpg-agent that uses the smart card daemon to access the keys on HSM:

1. Add the following text to the /root/.gnupg/gpg-agent.conf file:

```
scdaemon-program /usr/local/bin/gnupg-pkcs11-scd
```

> **NOTE:** If the /root/.gnupg/gpg-agent.conf file isn't available, then create the file and add the above lines.

2. Add/modify the following text to the /root/.gnupg/gnupg-pkcs11-scd.conf file:

```
providers p1

provider-p1-library /usr/safenet/lunaclient/lib/libCryptoki2_64.so
```

> **NOTE:** If gnupg-pkcs11-scd.conf file doesn't exist, you need create this file and copy all the contents from /usr/local/etc/gnupg-pkcs11-scd.conf.example or /usr/local/share/doc/gnupg-pkcs11-scd/gnupg-pkcs11-scd.conf.

3. Set the following environment variables to use the installed GPG:

```
# export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH

# export PATH=/usr/local/bin:$PATH
```

4. Execute the following command to connect the agent to HSM and get the keys from HSM:

```
# gpg-agent --server gpg-connect-agent
```

5. At the prompt, enter SCD LEARN. The pinentry program pop ups and prompts for the partition password. The output of the command will be similar to the following illustration:

NOTE: If you open the persistent session via salogin, the password prompt will not appear.

6. Look for the line S KEY-FRIENDLY, identify the signing/encryption/authentication certificate by the appropriate Common Name (CN), and copy the 20-byte SHA-1 hash in the gnupg-pkcs11-scd.conf file:

```
openpgp-sign 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000
openpgp-encr 7990A0D320B59A0DA525CE39D15398743762EFBB
openpgp-auth 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD
```

7. Use the following command to enable GPG to discover all useful information of the HSM partition:

```
# gpg --card-status
```

```
[root@localhost ~]# /usr/bin/gpg --card-status
Application ID ...: D2760001240111503131798 8A0061111
Version ..........: 11.50
Manufacturer .....: unknown
Serial number ....: 7988A006
Name of cardholder: [not set]
Language prefs ...: [not set]
Sex ..............: unspecified
URL of public key : [not set]
Login data .......: [not set]
Signature PIN ....: forced
Key attributes ...: 1R 1R 1R
Max. PIN lengths .: 0 0 0
PIN retry counter : 0 0 0
Signature counter : 0
Signature key ....: 8C5C E31F 726F E84C BB08  91E0 E281 6F2E F07F 0000
Encryption key....: 7990 A0D3 20B5 9A0D A525  CE39 D153 9874 3762 EFBB
Authentication key: 8B91 705A 7B3E D221 AAFF  5E78 B95C 89DD 4EB0 DDCD
General key info..: [none]
[root@localhost ~]#
```

8. Execute the following commands to generate the GPG virtual keys. Note that the keys are not actually generated on the local host and only a reference to the HSM keys is returned and registered by GPG.

```
# gpg --card-edit
```

```
# Command> admin
```

```
# Command> generate
```

9. Provide the following responses:

   a. Enter “y” to `Replace existing keys.`

```
Replace existing keys? (y/N) y
```

   b. Set the expiry parameter and then enter “y”.

```
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y
```

   c. Enter the real name and email address. Enter “O” to confirm.

```
GnuPG needs to construct a user ID to identify your key.

Real name: hsm@testgpg.com
Email address: hsm@testgpg.com
Comment:
You selected this USER-ID:
    "hsm@testgpg.com <hsm@testgpg.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
```

   Note this name as it will be used to reference the GPG and container image signing key going forward.

This completes the Integration of GnuPG with Thales Luna HSM.

OpenShift Container Platform Image Signing

To use trusted images for deploying application in OpenShift Container Platform, you need to perform the following tasks:

> Sign the container image

> Enable signature verification for container registries

> Verify container image signatures in OpenShift Container Platform

Sign the container image

Log in to the container registry, sign the container image using Skopeo, and then copy it to the container registry.

```
# skopeo copy --sign-by hsm@testgpg.com docker-daemon:signed_image:latest
docker://docker.io/<username>/signed_image:latest
```

```
[root@aa3533 ~]# skopeo copy --sign-by hsm@testgpg.com docker-daemon:signed_image:latest
docker://docker.io/type4ranjan/signed_image:latest
Getting image source signatures
Copying blob 4b8db2d7f35a done
Copying blob eeb14ff930d4 done
Copying blob f0f30197ccf9 done
Copying blob 431f409d4c5a done
Copying blob 02c055ef67f5 done
Copying blob c9732df61184 done
Copying config f0b8a9a541 done
Writing manifest to image destination
Signing manifest
Storing signatures
[root@aa3533 ~]#
```

NOTE: You can use any container registry.

Enable signature verification for container registries

1. Copy the signature from /var/lib/atomic/sigstore directory to any web server accessible to the OpenShift Container Platform.
2. Create a directory for config files.

   ```
   # mkdir worker_config_files
   ```

3. Change directory to worker_config_files.

   ```
   # cd worker_config_files
   ```

4. Create a public key titled signer-key.pub that will be used for image verification.

   ```
   # gpg2 --armor --export --output signer-key.pub hsm@testgpg.com
   ```

5. Create a file policy.json and add the following text to it:

```
{
    "default": [
        {
            "type": "insecureAcceptAnything"
        }
    ],
```

```
    "transports": {
        "docker": {
            "docker.io": [
                {
                    "keyType": "GPGKeys",
                    "type": "signedBy",
                    "keyPath": "/etc/pki/imagesigning/signer-key.pub"
                }
            ]
        },
        "docker-daemon": {
            "": [
                {
                    "type": "insecureAcceptAnything"
                }
            ]
        }
    }
}
```

6. Create a file default.yaml and add the following text to it:

```
default-docker:
  sigstore-staging: file:///var/lib/containers/sigstore

docker:
   docker.io:
    sigstore: http://10.124.138.126:8080/signatures
```

Here, 10.124.138.126 is the ip of the webserver where signatures are stored.

7. Change all the above files into base64 encoded format and export them to variables.

```
# export POLICY_CONFIG=$( cat policy.json | base64 -w0 )
# export REG_CONFIG=$( cat default.yaml | base64 -w0 )
# export SIGNER_KEY=$( cat signer-key.pub | base64 -w0)
```

8. Create a machine config file 51-worker-registry-trust.yaml using the cat command.

```
cat > 51-worker-registry-trust.yaml <<EOF
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 51-worker-registry-trust
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.1.0
    storage:
      files:
        - contents:
            source: >-
              data:text/plain;charset=utf-8;base64,${SIGNER_KEY}
          mode: 0644
          overwrite: true
          filesystem: root
```

```
          path: /etc/pki/imagesigning/signer-key.pub
        - contents:
            source: >-
              data:text/plain;charset=utf-8;base64,${REG_CONFIG}
          mode: 0644
          overwrite: true
          filesystem: root
          path: /etc/containers/registries.d/default.yaml
        - contents:
            source: >-
              data:text/plain;charset=utf-8;base64,${POLICY_CONFIG}
          mode: 0644
          overwrite: true
          filesystem: root
          path: /etc/containers/policy.json
EOF
```

9. Log in to the OpenShift Container Platform.

   ```
   # oc login
   ```

10. Select or create the project under which you want to deploy the application with trusted image. For example:

    ```
    # oc project lunaproject
    ```

11. Deploy the machine config.

    ```
    # oc create -f 51-worker-registry-trust.yaml
    ```

12. Verify that all the worker nodes in machine config pool are updated.

    ```
    # oc get mcp
    ```

```
[root@helper worker_config_files]# oc get mcp
NAME     CONFIG                                             UPDATED   UPDATING
DEGRADED   MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINE
master   rendered-master-c4cad0cda73631d0cacc0c676c439db6   True      False
False      3             3                   3
worker   rendered-worker-7185c747bb2053a5435c7d95ebedb625   True      False
False      3             3                   3
```

Verify container image signatures in OpenShift Container Platform

1. Create an application from any untrusted image.

   ```
   # oc new-app docker.io/<username>/unsigned_image
   ```

   Here, unsigned_image is any image that is not signed.

   Verify that the pod is not running.

   ```
   # oc get pods
   ```

```
[root@helper worker_config_files]# oc get pods
NAME                              READY   STATUS         RESTARTS   AGE
unsignedimage-76b59c4986-flvk9   0/1     ErrImagePull   0          5s
```

   Verify from logs that the image pull for untrusted image has failed.

   ```
   # oc get events | grep -i unsignedimage-76b59c4986-flvk9
   ```

```
[root@helper worker_config_files]# oc get events | grep -i unsignedimage-76b59c4986-flvk9
26s         Normal    AddedInterface           pod/unsignedimage-76b59c4986-flvk9    Add eth0 [10.253.2.14/23]
13s         Normal    Pulling                  pod/unsignedimage-76b59c4986-flvk9    Pulling image
"docker.io/type4ranjan/unsigned_image@sha256:ec0a51e3a052d846de03269189aaa1d387e3ab2f1e0244c0db0c731047b8555c"
25s         Normal    Scheduled                pod/unsignedimage-76b59c4986-flvk9    Successfully assigned
default/unsignedimage-76b59c4986-flvk9 to worker0.ocp4.example.com
12s         Warning   Failed                   pod/unsignedimage-76b59c4986-flvk9    Failed to pull image
"docker.io/type4ranjan/unsigned_image@sha256:ec0a51e3a052d846de03269189aaa1d387e3ab2f1e0244c0db0c731047b8555c": rpc error:
code = Unknown desc = Source image rejected: A signature was required, but no signature exists
12s         Warning   Failed                   pod/unsignedimage-76b59c4986-flvk9    Error: ErrImagePull
25s         Normal    BackOff                  pod/unsignedimage-76b59c4986-flvk9    Back-off pulling image
"docker.io/type4ranjan/unsigned_image@sha256:ec0a51e3a052d846de03269189aaa1d387e3ab2f1e0244c0db0c731047b8555c"
25s         Warning   Failed                   pod/unsignedimage-76b59c4986-flvk9    Error: ImagePullBackOff
24s         Normal    SuccessfulCreate         replicaset/unsignedimage-76b59c4986   Created pod:
unsignedimage-76b59c4986-flvk9
[root@helper worker_config_files]#
```

2. Create an application from trusted image.

```
# oc new-app docker.io/<username>/signed_image
```

3. Verify from logs that the image pull for trusted image has successfully completed.

```
# oc get events | grep -i signedimage-64c696cf45-8g9xh
```

```
[root@helper worker_config_files]# oc get events | grep -i signedimage-64c696cf45-8g9xh
18s         Normal    AddedInterface           pod/signedimage-64c696cf45-8g9xh      Add eth0 [10.253.2.20/23]
17s         Normal    Pulling                  pod/signedimage-64c696cf45-8g9xh      Pulling image
"docker.io/type4ranjan/signed_image@sha256:ec0a51e3a052d846de03269189aaa1d387e3ab2f1e0244c0db0c731047b8555c"
16s         Normal    Scheduled                pod/signedimage-64c696cf45-8g9xh      Successfully assigned
default/signedimage-64c696cf45-8g9xh to worker0.ocp4.example.com
11s         Normal    Pulled                   pod/signedimage-64c696cf45-8g9xh      Successfully pulled image
"docker.io/type4ranjan/signed_image@sha256:ec0a51e3a052d846de03269189aaa1d387e3ab2f1e0244c0db0c731047b8555c" in
6.949184192s
10s         Normal    Created                  pod/signedimage-64c696cf45-8g9xh      Created container signedimage
10s         Normal    Started                  pod/signedimage-64c696cf45-8g9xh      Started container signedimage
16s         Normal    SuccessfulCreate         replicaset/signedimage-64c696cf45     Created pod:
signedimage-64c696cf45-8g9xh
[root@helper worker_config_files]#
```

4. Verify that the pod is running.

```
# oc get pods
```

```
[root@helper worker_config_files]# oc get pods
NAME                              READY   STATUS    RESTARTS   AGE
signedimage-64c696cf45-8g9xh      1/1     Running   0          32s
```

This completes the integration of OpenShift Container Platform Image Signing with Luna HSM.

Contacting Customer Support

If you encounter a problem at any stage during this integration, contact Thales Customer Support. Thales Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Thales and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Customer Support Portal

The Customer Support Portal at https://supportportal.thalesgroup.com is a database where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable repository of support resources, including software and firmware downloads, release notes listing known problems and workarounds, a knowledge base, FAQs, product documentation, technical notes, and more. You can also use the portal to create and manage support cases.

> **NOTE:** You require an account to access the Customer Support Portal. To create a new account, go to the portal and click on the **REGISTER** link.

Telephone Support

If you have an urgent problem, or cannot access the Customer Support Portal, you can contact Thales Customer Support by telephone at +1 410-931-7520. Additional local telephone support numbers are listed on the support portal.

Email Support

You can also contact technical support by email at technical.support.DIS@thalesgroup.com.