THALES

gemalto
a Thales company

Container Image Signing

INTEGRATION GUIDE

SAFENET LUNA HSM

SAFENET DATA PROTECTION ON DEMAND

Document Information

Document Part Number	007-000636-001
Release Date	9 April 2020

Revision History

Revision	Date	Reason
A	9 April 2020	First Release

Gemalto does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms' resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Gemalto be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Gemalto products. Gemalto disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service or loss of privacy.

CONTENTS

PREFACE

This guide contains the following chapters:

- Getting Started describes the third party applications, supported platforms, prerequisites, and instructions related to the integration.
- Integrating GnuPG with SafeNet Luna HSM or DPoD describe the steps involved in Integrating GnuPG with SafeNet Luna HSM or DPoD.
- Setting up Image Signing with Docker describes the steps involved in configuring the host to verify signature, signing a container image, and trusting and validating signed images.
- Setting up Image Signing with OpenShift Container Platform describes the steps involved in configuring the host to verify signature, assigning signer and auditor roles to user, signing container image, verifying image signature, and pulling signed images.

Audience

This guide is intended for security administrators who are responsible for integrating GnuPG with SafeNet Luna HSM or HSM on Demand Service and then setting up image signing with Docker or OpenShift Container Platform.

All products manufactured and distributed by Gemalto, Inc. are designed to be installed, operated, and maintained by personnel who have the knowledge, training, and qualifications required to safely perform the tasks assigned to them. The information, processes, and procedures contained in this document are intended for use by trained and qualified personnel only.

Document Conventions

This section provides information on the conventions used in this document.

Notes

Notes are used to alert you to important or helpful information.

> **NOTE:** Take note. Notes contain important or helpful information.

Cautions

Cautions are used to alert you to important information that may help prevent unexpected results or data loss.

> **CAUTION!** Exercise caution. Caution alerts contain important information that may help prevent unexpected results or data loss.

Warnings

Warnings are used to alert you to the potential for catastrophic data loss or personal injury.

****WARNING** Be extremely careful and obey all safety and security measures. In this situation you might do something that could result in catastrophic data loss or personal injury**

Command Syntax and Typeface Conventions

Convention	Description
Bold	The bold attribute is used to indicate the following: > Command-line commands and options (Type dir /p.) > Button names (Click Save As.) > Check box and radio button names (Select the Print Duplex check box.) > Window titles (On the Protect Document window, click Yes.) > Field names (User Name: Enter the name of the user.) > Menu names (On the File menu, click Save.) (Click Menu > Go To > Folders.) > User input (In the Date box, type April 1.)
Italic	The italic attribute is used for emphasis or to indicate a related document. (See the *Installation Guide* for more information.)
Double quote marks	Double quote marks enclose references to other sections within the document.
<variable>	In command descriptions, angle brackets represent variables. You must substitute a value for command line arguments that are enclosed in angle brackets.
[optional] [<optional>]	Square brackets enclose optional keywords or <variables> in a command line description. Optionally enter the keyword or <variable> that is enclosed in square brackets, if it is necessary or desirable to complete the task.
[a | b | c] [<a> | <b> | <c>]	Square brackets enclose optional alternate keywords or variables in a command line description. Choose one command line argument enclosed within the braces, if desired. Choices are separated by vertical (OR) bars.
{ a | b | c } { <a> | <b> | <c> }	Braces enclose required alternate keywords or <variables> in a command line description. You must choose one command line argument enclosed within the braces. Choices are separated by vertical (OR) bars.

Customer Support

Gemalto 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 Gemalto and your organization. Please consult the support plan for further information about your entitlements, including the hours when telephone support is available to you.

Support Portal

The Customer Support Portal, at https://supportportal.thalesgroup.com, is a repository where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable database 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.

CHAPTER 1: Getting Started

This chapter covers the following topics:

About SafeNet Luna HSM and SafeNet DPoD

SafeNet 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. SafeNet Luna HSMs physically and logically secure cryptographic keys and accelerate cryptographic processing. The SafeNet Luna HSM on premise offerings include the SafeNet Luna Network HSM, SafeNet PCIe HSM, and SafeNet Luna USB HSMs.

SafeNet DPoD is a cloud-based platform that provides on-demand HSM and Key Management services through a simple graphical user interface. With DPoD, security is simple, cost effective, and easy to manage because there is no hardware to buy, deploy, and maintain. As an Application Owner, you click and deploy services, generate usage reports, and maintain only the services that you need.

Third party applications used

This integration uses the following third party applications:

- GnuPG (GPG version 2.0.22)
- Docker (version 1.13.1)
- Atomic (version 1.22.1)
- OpenShift Container Platform (version 3.11)

Supported operating systems

- For Integrating Docker with SafeNet Luna HSM and DPoD: The integration has been tested on RHEL 7.7, but it is also supported on RHEL 7.4 or above and RHEL Atomic Host 7.4 or above.
- For Integrating OpenShift Container Platform with SafeNet Luna HSM: The integration has been tested on RHEL 7.7 but it is also supported on RHEL 7.4 or above and RHEL Atomic Host 7.5 or above.

Prerequisites

Complete the following steps before you begin the integration:

Configure the SafeNet Luna Network HSM

If you are using a SafeNet Luna Network HSM, complete the following:

1. Verify the HSM is set up, initialized, provisioned, and ready for deployment. Refer to *SafeNet Luna Network HSM Product Documentation* for details.
2. Create a partition on the HSM that will be later used by Container Image Signing.
3. Register a client for the system and assign the client to the partition to create an NTLS connection. Initialize the Crypto Officer and Crypto User roles for the registered partition.
4. Ensure that each partition is successfully registered and configured. The command to see the registered partitions is:

```
# /usr/safenet/lunaclient/bin/lunacm
lunacm (64-bit) v7.3.0-165. Copyright (c) 2018 SafeNet. All rights reserved.

Available HSMs:

Slot Id ->              0
Label ->                gpgpartition
Serial Number ->        1280780175949
Model ->                LunaSA 7.3.0
Firmware Version ->     7.3.0
Configuration ->        Luna User Partition With SO (PW) Key Export With
Cloning Mode
Slot Description ->     Net Token Slot
Current Slot Id: 0
```

> **NOTE:** Follow the *SafeNet Luna Network HSM Product Documentation* for detailed steps for creating the NTLS connection, initializing the partitions, and initializing the Security Officer, Crypto Officer, and Crypto User roles.

Provision the HSM on Demand service

This service enables your client machine to access an HSM application partition for storing cryptographic objects used by your applications. Application partitions can be assigned to a single client, or multiple clients can be assigned to and share a single application partition.

You need to provision your application partition by initializing the following roles:

- **Security Officer (SO)** - Responsible for setting the partition policies and for creating the Crypto Officer role.
- **Crypto Officer (CO)** - Responsible for creating, modifying, and deleting crypto objects within the partition. The CO can use the crypto objects and create an optional, limited-capability role called Crypto User that can use the crypto objects but cannot modify them.

- **Crypto User (CU)** – An optional role that can use crypto objects while performing cryptographic operations.

> **NOTE:** Refer to the *SafeNet Data Protection on Demand Application Owner Quick Start Guide* for more information about provisioning the HSM on Demand service and creating a service client.
>
> The HSMoD service client package is a zip file that contains the software and configurations needed to connect your client machine to an existing HSM on Demand service.

Constraints on HSMoD Service

Please take the following limitations into consideration when integrating your application software with an HSM on Demand Service.

- **HSM on Demand Service in FIPS mode:** HSMoD service operates in a FIPS and non-FIPS mode. The FIPS mode is enabled by default. If your organization requires non-FIPS algorithms for your operations, ensure you enable the **Allow non-FIPS approved algorithms** check box when configuring your HSM on Demand service. Refer to the “Mechanism List” in the *SDK Reference Guide* for more information about the available FIPS and non-FIPS algorithms.
- **Verifying HSM on Demand <slot> value:** LunaCM commands work on the current slot. If you are completing an integration using HSMoD service, you need to verify the slot where you have sent the commands. If there is more than one slot, then use the **slot set** command to direct a command to a specified slot. You can use the slot list to map the slot numbers with HSMoD service.

Installing GPG-Dependent Packages

Before you begin the integration process, you need to install the following GPG-dependent packages from https://www.gnupg.org/download/index.html:

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

Installing Pinentry Package

To authenticate partition access for GPG, you can use the **salogin** utility, which gets automatically installed along with the SafeNet 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.

Installing GPG Package

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

Installing PKCS11-Helper and GnuPG-PKCS11 SCD

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

> **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.

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

Installing Atomic,Docker and Openshift Conatiner Platform

Install Atomic,Docker and OpenShift Container Platform by:

- **Atomic:** To install Atomic, use the command: `yum install atomic`
- **Docker:** To install Docker, please refer https://docs.docker.com/install/
- **OpenShift Container Platform:** To install OpenShift Container Platform, refer to https://docs.openshift.com/

> **NOTE:** You don’t need to install OpenShift Conatiner Platform in case you are integrating Docker with SafeNet Luna HSM or DPoD.

CHAPTER 2: Integrating GnuPG with SafeNet Luna HSM or DPoD

To Integrate GnuPG with SafeNet Luna HSM or DPoD complete the following:

> Accessing SafeNet HSM
> Configuring the gnupg-pcs11-scd.conf file
> Generating Keys and Certificates
> Configuring GPG to use the PKCS#11 Smart Card Daemon

Accessing SafeNet Luna HSM or DPoD

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

> Using salogin utility
> Using Pinentry

Using salogin utility

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

> **NOTE:** Persistent Session is not supported in DPoD so salogin will not work. For DPoD go to Using Pinentry section.

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` represent the slot_id and `-i` represent the AppId set in the Chrystoki.conf file.

Using 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 doesn't exist, you need to create it at /root/.gnupg/ directory.

Configuring 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:

1. Add/modify the following lines to /root/.gnupg/gnupg-pkcs11-scd.conf file

```
provider-p1-allow-protected-auth

provider-p1-cert-private

provider-p1-private-mask 0
```

> **NOTE**: If the 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.

Generating Keys and Certificates

After creating the NTLS connection with HSM, follow the below steps to generate the RSA key pair on HSM. GPG uses several asymmetric key pairs as part of the 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 SafeNet 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

> **NOTE:** CMU command option might be slightly differ in other versions of Luna Client, kindly refer to the Luna SA documentation for exact options.

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 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.

Configuring 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 line to the /root/.gnupg/gpg-agent.conf file:

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

> **NOTE:** if **/root/.gnupg/gpg-agent.conf** file is not present, then create the file and add the above lines.

2. Add/modify the following lines to THE /root/.gnupg/gnupg-pkcs11-scd.conf file available at the :

```
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:

```
[root@localhost ~]# /usr/bin/gpg-agent --server gpg-connect-agent
OK Pleased to meet you
SCD LEARN
gnupg-pkcs11-scd[25660.1322522368]: Listening to socket '/tmp/gnupg-pkcs11-scd.FXhPEL/agent.S'
gnupg-pkcs11-scd[25660.1322522368]: accepting connection
gnupg-pkcs11-scd[25660]: chan_0 -> OK PKCS#11 smart-card server for GnuPG ready
gnupg-pkcs11-scd[25660.1322522368]: processing connection
gnupg-pkcs11-scd[25660]: chan_0 <- GETINFO socket_name
gnupg-pkcs11-scd[25660]: chan_0 -> D /tmp/gnupg-pkcs11-scd.FXhPEL/agent.S
gnupg-pkcs11-scd[25660]: chan_0 -> OK
gnupg-pkcs11-scd[25660]: chan_0 <- OPTION event-signal=12
gnupg-pkcs11-scd[25660]: chan_0 -> OK
gnupg-pkcs11-scd[25660]: chan_0 <- LEARN
gnupg-pkcs11-scd[25660]: chan_0 -> S SERIALNO D27600012401115031317988A0061111
S SERIALNO D27600012401115031317988A0061111
gnupg-pkcs11-scd[25660]: chan_0 -> S APPTYPE PKCS11
S APPTYPE PKCS11
gnupg-pkcs11-scd[25660]: chan_0 -> INQUIRE NEEDPIN PIN required for token 'deepak' (try 0)
gnupg-pkcs11-scd[25660]: chan_0 <- [ 44 20 74 65 6d 70 31 32 33 23 00 00 00 00 00 00 ...(76 byte(s) skipped) ]
gnupg-pkcs11-scd[25660]: chan_0 <- END
gnupg-pkcs11-scd[25660]: chan_0 -> S KEY-FRIEDNLY 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000 /C=IN/ST=UPST/L=NOIDA/O=GEMALTO/OU=IDSS/CN=GPG-Sign on deepak
S KEY-FRIEDNLY 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000 /C=IN/ST=UPST/L=NOIDA/O=GEMALTO/OU=IDSS/CN=GPG-Sign on deepak
gnupg-pkcs11-scd[25660]: chan_0 -> S KEY-FPR 1 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000
S KEY-FPR 1 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000
gnupg-pkcs11-scd[25660]: chan_0 -> S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110001
S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110001
gnupg-pkcs11-scd[25660]: chan_0 -> S KEYPAIRINFO 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110001
S KEYPAIRINFO 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110001
gnupg-pkcs11-scd[25660]: chan_0 -> S KEY-FRIEDNLY 7990A0D320B59A0DA525CE39D15398743762EFBB /C=IN/ST=UPST/L=NOIDA/O=GEMALTO/OU=IDSS/CN=GPG-Encr on deepak
S KEY-FRIEDNLY 7990A0D320B59A0DA525CE39D15398743762EFBB /C=IN/ST=UPST/L=NOIDA/O=GEMALTO/OU=IDSS/CN=GPG-Encr on deepak
gnupg-pkcs11-scd[25660]: chan_0 -> S KEY-FPR 2 7990A0D320B59A0DA525CE39D15398743762EFBB
S KEY-FPR 2 7990A0D320B59A0DA525CE39D15398743762EFBB
gnupg-pkcs11-scd[25660]: chan_0 -> S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110010
S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110010
gnupg-pkcs11-scd[25660]: chan_0 -> S KEYPAIRINFO 7990A0D320B59A0DA525CE39D15398743762EFBB Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110010
S KEYPAIRINFO 7990A0D320B59A0DA525CE39D15398743762EFBB Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110010
gnupg-pkcs11-scd[25660]: chan_0 -> S KEY-FRIEDNLY 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD /C=IN/ST=UPST/L=NOIDA/O=GEMALTO/OU=IDSS/CN=GPG-Auth on deepak
S KEY-FRIEDNLY 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD /C=IN/ST=UPST/L=NOIDA/O=GEMALTO/OU=IDSS/CN=GPG-Auth on deepak
gnupg-pkcs11-scd[25660]: chan_0 -> S KEY-FPR 3 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD
S KEY-FPR 3 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD
gnupg-pkcs11-scd[25660]: chan_0 -> S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110011
S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110011
gnupg-pkcs11-scd[25660]: chan_0 -> S KEYPAIRINFO 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110011
S KEYPAIRINFO 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110011
gnupg-pkcs11-scd[25660]: chan_0 -> OK
OK
```

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 as follows:

```
openpgp-sign 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000

openpgp-encr 7990A0D320B59A0DA525CE39D15398743762EFBB

openpgp-auth 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD
```

7. Use the following command to enable GPG to discover all useful information of the card (or HSM partition in this case):

```
# 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
   ```

   You need to provide the following inputs:

   a. Respond “y” to Replace existing keys?

   b. Do not backup keys if prompted.

   c. Set the expiry parameter

   d. Provide the key name when prompted for Real name.

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

This complete the Integration of GnuPG with SafeNet Luna HSM or DPoD.

CHAPTER 3: Setting up Image Signing with Docker

Follow these steps to set up image signing with Docker after you've integrated GPG with SafeNet Luna HSM or DPoD:

> Configuring the host to verify signature and export GPG public key

> Signing a container image

> Trusting and validating signed images

Configuring the host to verify signature and export GPG public key

NOTE: In this document, the GPG key is labeled as hsm@testgpg.com and the Docker image is labeled as testimage.

For configuring the host to verify signature and export GPG Public Key:

1. Open the /etc/sysconfig/docker file and change **--signature-verification=false** to **--signature-verification=true** in the **OPTIONS**:

   ```
   # OPTIONS='--selinux-enabled --log-driver=journald --signature-verification=true'
   ```

2. Restart docker to load the changes.

   ```
   # systemctl restart docker
   ```

3. List the secret keys.

   ```
   # gpg --list-secret-keys
   ```

4. Create the public key that will be used for image verification.

   ```
   # gpg --armor --export --output /etc/pki/containers/key.pub hsm@testgpg.com
   ```

Signing a container image

You can adopt either of the two approaches to sign the container image:

> Creating an image signature for an image in a registry

> Creating an image signature at push time

Creating an image signature for an image in a registry

To create an image signature for an image that is already pushed to a registry:

1. Find the image for which you want to create the signature using the docker search command:

```
# docker search <username>/testimage
```

2. Login to the registry

```
# docker login
```

Provide the username and the password

3. Sign the image

```
# atomic sign  --sign-by hsm@testgpg.com --gnupghome /root/.gnupg/
<username>/testimage
```

```
[root@localhost 64]# atomic sign  --sign-by hsm@testgpg.com --gnupghome /root/.gnupg/
<username>/testimage
Trying docker.io/<username>/testimage:latest...
Created:
/var/lib/atomic/sigstore/<username>/testimage@sha256=25e4971b0f640e2e67b51cfa5e129c35ff1149f8c3056
a58b9258bad6ed5c5f7/signature-1
```

The signature is created and stored in the **/var/lib/atomic/sigstore** directory on the local system under the registry name, user name, and image name.

Creating an image signature at push time

To create an image signature at push time:

1. Use the image ID to tag the image with the identity of the registry.

```
# docker tag testimage docker.io/<username>/testimage:latest
```

2. Login to the registry.

```
# docker login
```

Provide the username and the password when prompted.

3. Push the image. The image is signed at push time.

```
# atomic push -t docker --sign-by hsm@testgpg.com --gnupghome /root/.gnupg
docker.io/<username>/testimage:latest
```

```
[root@localhost 64]# atomic push -t docker --sign-by hsm@testgpg.com --gnupghome /root/.gnupg
docker.io/<username>/testimage:latest
Copying blob 77b174a6a187 done
Copying blob 474d3923bd04 done
Copying config 23c02f835c done
Writing manifest to image destination
Signing manifest
```

When prompted, enter the partition password. At this point, the image should be available from the repository and ready to pull.

Adding default signer for every images (optional)

You can also add default signing information to /etc/atomic.conf file that will be used every time you use atomic push or atomic sign. To add default signer for every image:

1. Add/Modify the following entries in /etc/atomic.conf file. In this document, the default_signer is labeled as hsm@testgpg.com and gnupg_homedir is labeled as /root/.gnupg.

```
default_signer: hsm@testgpg.com
```

```
gnupg_homedir: /root/.gnupg
```

2. Login to the registry.

   ```
   # docker login
   ```

   Provide username and password when prompted.

3. Sign the image using either of the two methods:

   - Sign the image in registry

     ```
     # atomic sign docker.io/<username>/testimage
     ```

     ```
     [root@localhost 64]# atomic sign <username>/testimage
     Trying docker.io/<username>/testimage:latest...
     Created:
     /var/lib/atomic/sigstore/<username>/testimage@sha256=25e4971b0f640e2e67b51cfa5e129c35ff1149f8c3056
     a58b9258bad6ed5c5f7/signature-2
     ```

   - Push the image to registry.

     ```
     # atomic push -t docker docker.io/<username>/testimage:latest
     ```

     ```
     [root@localhost 64]# atomic push -t docker docker.io/<username>/testimage:latest
     Copying blob 474d3923bd04 done
     Copying blob 77b174a6a187 done
     Copying config 23c02f835c done
     Writing manifest to image destination
     Signing manifest
     Storing signatures
     ```

   When prompted, enter the partition password. At this point, the image should be available from the repository and ready to pull.

Trusting and validating signed images

Follow these steps to trust and validate signed images:

1. Remove the images if they already exist so that new images can be pulled.

   ```
   # docker rmi docker.io/<username>/testimage
   ```

   ```
   # docker rmi testimage
   ```

2. Check the current trust value for pulling container images with the **atomic** trust command.

   ```
   # atomic trust show
   ```

3. Set default value to reject all the images.

   ```
   # atomic trust default reject
   ```

   ```
   # atomic trust show
   ```

4. Add trusted registry with signatures.

   ```
   # atomic trust add  --pubkeys /etc/pki/containers/key.pub --sigstore
   file:///var/lib/atomic/sigstore docker.io/<username>
   ```

   Alternatively, you can perform the following steps:

   e. Open the /etc/containers/policy.json file and add the following changes.

   ```
   "default": [
   ```

```
            {
                "type": "reject"
            }
        ],
        "transports": {
            "docker": {
                "registry.access.redhat.com": [
                    {
                        "type": "insecureAcceptAnything"
                    }
                ],
                "docker.io/<username>": [
                    {
                        "keyType": "GPGKeys",
                        "type": "signedBy",
                        "keyPath": "/etc/pki/containers/key.pub"
                    }
                ]
            }
        }
    }
```

f. Create a file /etc/containers/registries.d/docker.io-<username>.yaml and add the **sigstore** for the registry docker:

```
docker.io/<username>:
  sigstore: file:///var/lib/atomic/sigstore
```

5. Check that the trusted registry is added.

```
# atomic trust show
```

```
[root@localhost ~]# atomic trust show
* (default)                reject
docker.io/<username>       signed hsm@testgpg.com   file:///var/lib/atomic/sigstore
```

6. Pull the image.

```
# atomic pull docker.io/<username>/testimage:latest
```

```
[root@localhost ~]# atomic pull docker.io/<username>/testimage
Pulling docker.io/<username>/testimage:latest ...
Copying blob ab5ef0e58194 done
Copying blob eeb389d1c9fb done
Copying blob a32d66072b52 done
Copying blob 567aa51b3d35 done
Copying config 88533e5d03 done
Writing manifest to image destination
Storing signatures
```

7. Check that the image is pulled successfully.

   ```
   # docker images
   ```

This completes the process of setting up image signing with Docker.

CHAPTER 4: Setting up Image Signing with OpenShift Container Platform

Follow these steps to set up image signing with OpenShift Container Platform after you've integrated GPG with SafeNet Luna HSM or DPoD:

> Configuring the host to verify signature and export GPG Public Key
> Assigning signer and auditor roles to user
> Signing container image
> Verifying image signature
> Pulling signed images from OpenShift registry to local docker-daemon

Configuring the host to verify signature and export GPG Public Key

To configure the host to verify signature and export GPG Public Key:

1. Open /etc/sysconfig/docker file and change **--signature-verification=false** to **--signature-verification=true** in the **OPTIONS**:

   ```
   OPTIONS='--selinux-enabled --log-driver=journald --signature-verification=true'
   ```

2. Restart docker to load the changes.

   ```
   # systemctl restart docker
   ```

3. List the secret keys.

   ```
   # gpg --list-secret-keys
   ```

   NOTE: In this document, the gpg key is labeled as hsm@testgpg.com and docker image is labeled as testimage.

4. Create the public key that will be used for image verification.

   ```
   # gpg --armor --export --output /etc/pki/containers/key.pub hsm@testgpg.com
   ```

Assigning signer and auditor roles to user

To be able to sign and verify the image the user must have the role for image-signer and image-auditor assigned.

1. To attach the signature to the image, assign image-signer cluster role to user.

   ```
   # oc adm policy add-cluster-role-to-user system:image-signer <user_name>
   ```

2. To verify the signature of an image, assign image-auditor cluster role to user.

   ```
   # oc adm policy add-cluster-role-to-user system: image-auditor <user_name>
   ```

Signing container image

To create an Image Signature at Push Time:

1. For the purpose of demonstration, a project **mytestproject** is created in OpenShift container.

   ```
   # oc new-project mytestproject
   ```

2. Check the **imagestreams** inside the OpenShift that are already present.

   ```
   # oc get imagestreams
   ```

3. Tag the docker image that you want to sign and push to OpenShift registry 172.30.1.1:5000.

   ```
   # docker tag  testimage 172.30.1.1:5000/mytestproject/testimage
   ```

4. List the docker images and check that tagged docker image exists.

   ```
   # docker images
   ```

5. Obtain the token of the user that is logged in.

   ```
   # oc whoami -t
   ```

6. Push the image to the OpenShift registry using **atomic** command.

   ```
   # atomic push --type atomic --sign-by hsm@testgpg.com
   172.30.1.1:5000/mytestproject/testimage:latest
   ```

   When prompted, provide username and token of the user obtained in step 5.

   ```
   [root@localhost ~]# atomic push --type atomic --sign-by hsm@testgpg.com
   172.30.1.1:5000/mytestproject/testimage:latest
   Registry username: system
   Registry password:

   Getting image source signatures
   Copying blob 77b174a6a187 done
   Copying blob 474d3923bd04 done
   Copying config 23c02f835c done
   Writing manifest to image destination
   Signing manifest
   Storing signatures
   ```

7. Check the **imagestreams** inside the OpenShift and verify that image has been pushed successfully.

   ```
   # oc get imagestreams
   ```

   ```
   [root@localhost ~]# oc get imagestreams
   NAME          DOCKER REPO                                    TAGS      UPDATED
   testimage   172.30.1.1:5000/mytestproject/testimage   latest    6 minutes ago
   ```

Verifying Image Signature

To verify the image signature in OpenShift container registry, you must perform the following steps:

1. Describe the **imagestream** and check its status as **Status: Unverified**.

   ```
   # oc describe istag testimage:latest -n mytestproject
   ```

2. Verify the image signature.

```
# oc adm verify-image-signature
sha256:f972d050f3c893bad0b1b9f875fefb39d6769e879e1bc191faea9b989bb038a7 --
expected-identity=172.30.1.1:5000/mytestproject/testimage:latest  --public-
key=/etc/pki/containers/key.pub --save
```

3. Describe the **imagestream** and check its status as **Status: Verified**.

```
# oc describe istag testimage:latest -n mytestproject
```

Pulling signed images from OpenShift registry to local host

To pull signed images from **OpenShift** registry to local docker-daemon:

1. Remove the images if they already exist.

```
# docker rmi 172.30.1.1:5000/mytestproject/testimage
```

```
# docker rmi testimage
```

2. Login to the **OpenShift** registry

```
# docker login 172.30.1.1:5000
```

Provide the username and token

3. Check the current trust value for pulling container images with the atomic command

```
# atomic trust show
```

4. Set default value to reject all the images

```
# atomic trust default reject
```

```
# atomic trust show
```

5. Add the trusted registry.

```
atomic trust add  --pubkeys /etc/pki/containers/key.pub
172.30.1.1:5000/mytestproject
```

Or alternatively, you can open the /etc/containers/policy.json file and make the following changes.

```
"default": [
        {
            "type": "reject"
        }
    ],
    "transports": {
        "docker": {
            "registry.access.redhat.com": [
                {
                    "type": "insecureAcceptAnything"
                }
            ],
```

```
        "172.30.1.1:5000/mytestproject": [
                {
                    "keyType": "GPGKeys",
                    "type": "signedBy",
                    "keyPath": "/etc/pki/containers/key.pub"
                }
            ]
        }
    }
}
```

6. Check that the trust is added.

```
# atomic trust show
```

```
[root@localhost ~]# atomic trust show
* (default)                          reject
172.30.1.1:5000/mytestproject       signed hsm@testgpg.com
```

7. Pull the image.

```
# atomic pull 172.30.1.1:5000/mytestproject/testimage:latest
```

```
[root@localhost ~]# atomic pull 172.30.1.1:5000/mytestproject/testimage:latest
Pulling 172.30.1.1:5000/mytestproject/testimage:latest ...
Copying blob 793c50de1bc6 done
Copying blob 56fc96aef6ff done
Copying config 23c02f835c done
Writing manifest to image destination
Storing signatures
```

8. Check that the image is pulled successfully.

```
# docker images
```

```
[root@localhost ~]# docker images
REPOSITORY                                  TAG                 IMAGE ID            CREATED             SIZE
172.30.1.1:5000/mytestproject/testimage     latest              23c02f835cfb        20 hours ago        203 MB
```

This completes the process of setting up image signing with OpenShift Container Platform.