Fortanix®

Integration Guide

FORTANIX DATA SECURITY MANAGER USING KERNEL MODULE SIGNING GUIDE

Fortanix, Inc. | 3910 Freedom Circle | Suite 104 | Santa Clara, CA 95052 | United States of America | +1 (650) 943-2484 |
info@fortanix.com | www.fortanix.com

TABLE OF CONTENTS

1.0 INTRODUCTION

Welcome to the Fortanix Data Security Manager (DSM) integration guide. This document provides step-by-step instructions on how to sign the Linux kernel module files using the sign-file utility, Fortanix DSM, and the Fortanix Public-Key Cryptography Standards (PKCS#11) client.

When utilizing an external Key Management Service (KMS) such as Fortanix, the sign-file utility offers two alternatives:

- Utilize the PKCS#11 URI to transmit the signing request to the remote key. Although this feature might not be accessible in older Linux kernel versions.
- Retrieve the signature from the remote key, save it in a file, and supply it as input to the utility to append it to the file.

This document covers the following:

- Configuring Fortanix DSM
- Setting up the PKCS#11 client in the signing server
- Creating kernel signing public certificates
- Signing kernel modules using the Fortanix DSM key
- Troubleshooting tips

2.0 TERMINOLOGY REFERENCES

- **Fortanix Data Security Manager** -

  Fortanix DSM is the cloud solution secured with Intel® SGX. With Fortanix DSM, you can securely generate, store, and use cryptographic keys and certificates, as well as secrets, such as passwords, API keys, tokens, or any blob of data.

- **Accounts** -

  A Fortanix DSM account is the top-level container for security objects managed by the Fortanix DSM. An account is generally associated with an organization, rather than an individual. Security objects, groups, and applications belong to exactly one account. Different accounts are fully isolated from each other. *See support for more information.*

- **Users** -

  Users are associated with an email address. A user can be a member of one or more accounts. Depending on permissions, users can:

  - Perform management operations like adding or modifying users or groups
  - Create security objects
  - Change properties of security objects
  - Review logs of Fortanix DSM activity

  ⚠ **Users cannot perform cryptographic operations. Only applications can perform cryptographic operations.**

- **Groups** -

  A group is a collection of security objects created by and accessible by users and applications which belong to the group. The user who creates a group automatically gets assigned the role of the Group Administrator. You can add more users to the group in the role of administrators or auditors. You can also add applications to the group to enable the applications to create and use security objects in that group. *See support for more information.*

  Access policies are set at the group level, so all security objects in a group share the same access policy. Any number of users and/or applications can be assigned to a group. *Some examples of usage of groups are given in the Authorization section*.

  Quorum policies can also be set at group level. A Quorum policy mandates that all security sensitive operations in that group would require a quorum approval. Such operations include using a key for cryptographic operations or deleting or updating a group. *See Quorum Policy for more information.*

- **Applications** -

  An application can use Fortanix DSM to generate, store, and use security objects, such as cryptographic keys, certificates, or an arbitrary secret. Applications can authenticate to Fortanix DSM using an API key (a secret token) or a TLS client certificate. An application can

interact with Fortanix DSM using the REST APIs or using the PKCS#11, JCE, or CNG providers. *See support for more information.*

- **Fortanix Data Security Manager Security Objects** –

  A security object is any datum stored in Fortanix DSM (for example a key, a certificate, a password, or other security objects). Each security object is assigned to exactly one group. users and applications assigned to the group have permission to see the security object and to perform operations on it. *See support for more information.*

3.0 CONFIGURING FORTANIX DSM

3.1 CREATE AN ACCOUNT

Ensure to create an account within the Fortanix Data Security Manager (DSM). *For detailed steps, refer to the Create an Account documentation.*

3.2 CREATE A GROUP

Ensure to create a group within the Fortanix Data Security Manager (DSM). *For detailed steps, refer to the Create a Group documentation.*

3.3 CREATE AN APPLICATION

Ensure to create an application within the Fortanix Data Security Manager (DSM). *For detailed steps, refer to the Create an Application documentation.*

You have the option to select either API key authentication or client certificate-based authentication. *For more details about application authentication methods, refer to the Authentication documentation.*

NOTE: Ensure to copy the API key of the created application for later use.

3.4 CREATE A SECURITY OBJECT

Generate an RSA key within the Fortanix DSM group created in *Section 3.2: Create a Group*. Ensure to select the **PKCS1v15** signature option for **Encryption** in the **Padding Policy** column. The key

must have a minimum set of permissions, including **SIGN**, **VERIFY**, and **APPMANAGEABLE**. You can select additional permissions according to your specific needs.

4.0 SET UP FORTANIX PKCS#11 CLIENT

This section illustrates the procedures for configuring the Fortanix PKCS#11 client on the signing server.

4.1 DOWNLOAD THE FORTANIX PKCS#11 CLIENT

Download the latest `.so` file for the *PKCS#11 client* and copy it to the signing server.

4.2 INTEGRATE THE FORTANIX PKCS#11 CLIENT

Perform the following steps for integrating the Fortanix PKCS#11 client with the signing server:

1. Install the OpenSSL PKCS#11 engine:
   a. For Debian-based Linux distributions (including Ubuntu), run the following command:

```
sudo apt install libengine-pkcs11-openssl
```

   b. For CentOS, RHEL, or Fedora (with EPEL repository available), run the following command:

```
yum install engine_pkcs11
```

   c. Run the following command to verify if the PKCS#11 engine is installed:

```
adminguy@ubuntu:~$ openssl engine pkcs11
(pkcs11) pkcs11 engine
```

2. Create an OpenSSL configuration file, such as `openssl.conf`, with the following content:

```
openssl_conf = openssl_def

[openssl_def]
engines = engine_section

[req]
```

```
distinguished_name = req_distinguished_name
[req_distinguished_name]
# empty.

[engine_section]
pkcs11 = pkcs11_section

[pkcs11_section]
engine_id = pkcs11
dynamic_path = /path/to/pkcs11 engine
MODULE_PATH = /path/to/fortanix pkcs11 file
init = 0
```

4.3 CREATE THE PKCS#11 CONFIGURATION FILE

Perform the following steps to create and configure the PKCS#11 file for the PKCS#11 client:

1. If using API key-based app authentication:

```
vi pkcs11.conf
api_endpoint = "https://apps.smartkey.io" # default is
"https://apps.smartkey.io"
api_key= "" # Paste the api key copied above here.

[log]
system = true # Unix only, logs to syslog
file = "/path/to/log/file"
```

2. Perform the following steps to create a self-signed certificate with a key in Fortanix DSM:
   a. Run the following command to set the Fortanix DSM App UUID:

```
export FORTANIX_APP_UUID=<Fortanix DSM App UUID>
```

Where, `<Fortanix DSM App UUID>` refers to the Fortanix DSM App UUID created in *Section 3.4: Create a Security Object*.

b. Run the following command to set the location of OpenSSL configuration file:

```
export OPENSSL_CONF=/location/of/openssl conf file
```

c. Run the following command to generate a self-signed certificate with key:

```
openssl req -newkey rsa:2048 -nodes -keyout private.key -x509  -
days 365 -out certificate.crt -subj
"/C=US/ST=California/L=Mountain View/O=Fortanix,
Inc./OU=SE/CN=$FORTANIX_APP_UUID"
```

3. Run the following OpenSSL command to convert the private key file to PKCS#8 `.pem` format:

```
openssl pkcs8 -topk8 -in private.key -out private.pem -nocrypt
```

This command will take the input private key file (`private.key`), convert it to PKCS#8 format, and save the result in the output file (`private.pem`). The `-nocrypt` option ensures that the conversion process does not encrypt the private key with a passphrase.

2. After creating the client certificate, upload this client certificate to the Fortanix DSM application. *For detailed steps, refer to the Import a Security Object documentation.*
3. Create a `/etc/fortanix` directory.
4. Create a PKCS#11 configuration file with the following details:

```
vi pkcs11.conf
api_endpoint = "https://apps.smartkey.io" # default is
"https://apps.smartkey.io"
cert_file = "/path/to/cert.pem" # X.509 PEM client certificate
key_file = "/path/to/key.pem" # PKCS#8 PEM client private key
app_id = "d82f4372-06d2-47e4-a064-d243bc029b64" #appuuid

[log]
system = true # Unix only, logs to syslog
file = "/path/to/log/file"
```

5.0 CREATE THE FORTANIX KERNEL SIGNING PUBLIC CERTIFICATE

Run the following commands to generate the kernel signing public certificate using the PKCS#11 URI from the key located in Fortanix:

```
export FORTANIX_PKCS11_NUM_SLOTS
export OPENSSL_CONF=/location/of/openssl/conf/file
openssl req -new -x509 -days 365 -sha256 -engine pkcs11 -keyform engine -
key pkcs11:object=KernelSigningRSA?pin-
value=file:///etc/fortanix/pkcs11.conf -outform der -out
Kernelsigningcert.der -subj "/CN=test.example.com"
```

Where, `KernelSigningRSA` refers to the actual key name in Fortanix. Ensure that the `pin-value` points to the location of the `pkcs11.conf` file.

NOTE:

- You can modify the subject parameter according to your specific requirements.
- While self-signed certificates are currently in use, you can use CA-signed certificates as well.

Alternatively, run the following command if PKCS#11 URI is not supported in your OpenSSL version:

```
openssl req -engine pkcs11 -keyform engine -new -key 1:<ID> -nodes -days
365 -x509 -sha256 -out test.pem -subj "/CN=test.example.com"
```

Where, `<ID>` refers to the Fortanix DSM key UUID created in *Section 3.4: Create a Security Object*.

6.0 SIGN THE FORTANIX KERNEL MODULES

6.1 SIGN USING PKCS#11 URI

Perform the following steps only if signing the kernel modules using the PKCS#11 URI.

Run the following command to invoke the sign-file utility for signing:

```
export FORTANIX_PKCS11_NUM_SLOTS=1
./sign-file sha256 <pkcs11 uri for key> <public certificate> <module file
for signing> <destination of signed file>
```

Where,

- `<pkcs11 uri for key>` refers to the PKCS#11 URI for the Fortanix DSM key.
- `<public certificate>` refers to the public certificate used for signing.
- `<module file for signing>` refers to the kernel module file that you want to sign.
- `<destination of signed file>` refers to the location where the signed module file will be saved.

For example,

```
./sign-file sha256 pkcs11:object=KernelSigningRSA?pin-
value=file:///etc/fortanix/pkcs11.conf /etc/fortanix/Kernelsigningcert.der
/etc/fortanix/mfe_aac_100713218.ko
/etc/fortanix/mfe_aac_100713218_signed.ko
```

6.2 SIGN USING SIGNATURE FILE

Perform the following steps only if signing the kernel modules using a signature file.

NOTE: Ensure that `jq` is installed on the system.

1. Run the following command to obtain the signature from Fortanix using a sample script:

```
hashbase64=`sha256sum hello.ko | awk '{printf $1}' | xxd -r -p |
base64`
json='{
  "hash_alg": "sha256",
  "hash": "",
  "mode": {
    "PKCS1_V15": {}
  },
  "deterministic_signature": true
}'
```

```
json=$( jq --arg value "$hashbase64" '.hash= $value' <<< "$json")
echo $json
signature=`curl --location --request POST
'https://<URL>/crypto/v1/keys/<key UUID>/sign' --header
'Authorization: Basic <API Key>' --data "$json" | jq -r
'.signature'`
echo -n  $signature | base64 -d > signature.raw
```

Where,

- `<hello.ko>` refers to the specific `.ko` file for which you want to obtain the signature.
- `<URL>` refers to the endpoint where you can access Fortanix DSM for signature retrieval.
- `<key UUID>` refers to the unique identifier assigned to your Fortanix DSM key.
- `<API Key>` refers to the authentication key needed for authorization in the curl request.

2. Run the following command to sign:

```
export FORTANIX_PKCS11_NUM_SLOTS=1
 ./sign-file -s <signature-file>  <public certificate> <module file
for signing> <destination of signed file>
```

Where,

- `<signature file>` refers to the file obtained in the *Step 1*.
- `<public certificate>` refers to the public certificate used for signing.
- `<module file for signing>` refers to the kernel module file that you want to sign.
- `<destination of signed file>` refers to the location where the signed module file will be saved.

For example,

```
sign-file -s /etc/fortanix/signature.raw sha256
/etc/fortanix/Kernelsigningcert.der /etc/fortanix/hello.ko
/etc/fortanix/hello_signed.ko.
```

7.0 TROUBLESHOOTING

If you encounter issues, refer to the log file specified in `pkcs11.conf` for any potential issues or errors.

8.0 DOCUMENT INFORMATION

8.1 DOCUMENT LOCATION

https://support.fortanix.com/hc/en-us/articles/23447625237908-Using-Fortanix-Data-Security-Manager-for-Kernel-Module-Signing-Module

8.2 DOCUMENT UPDATES

This document will typically be updated on a periodic review and update cycle.

For any urgent document updates, please send an email to: support@fortanix.com