Configure IOx Package Signature Validation

Contents

Introduction

This document describes in a detailed way how to create and use signed packages on the IOx platform.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Basic Linux knowledge
- Understand how certificates work

Components Used

The information in this document is based on these software and hardware versions:

- IOx capable device that is configured for IOx:
  IP address configuredGuest Operating System (GOS) and Cisco Application Framework (CAF) that runsNetwork Address Translation (NAT) configured for access to CAF (port 8443)
- Linux host with open Secure Sockets Layer (SSL) installed
- IOx client installation files which can be downloaded from: https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Background Information

Since IOx release, AC5 application package signing is supported. This feature allows to ensure that the application package is valid and the one installed on the device is obtained from a trusted source. If Application Package Signature Validation is turned ON in a platform, only then signed applications can be deployed.

Configure

These steps are required to use package signature validation:

1. Create a Certificate Authority (CA) key and certificate.
2. Generate a trust anchor for use on IOx.
3. Import the trust anchor on your IOx-device.
4. Create an application specific key and Certificate Signing Request (CSR).
5. Sign the application specific certificate with the use of the CA.
6. Package your IOx application, sign it with the application specific certificate.
7. Deploy your signed IOx package onto a signature-enabled device.

Note: For this article, a self-signed CA is used in a production scenario. The best option is to use an official CA or your company's CA to sign.

Note: The options for the CA, keys and signatures are chosen for lab purposes only and might need to be adjusted for your environment.

Step 1. Create CA Key and Certificate

The first step is to create your own CA. This can be simply done by the generation of a key for the CA and a certificate for that key:

In order to generate the CA key:

```
[jedepuyd@KJK-SRVIOT-10 signing]$ openssl genrsa -out rootca-key.pem 2048
Generating RSA private key, 2048 bit long modulus
.....................................................................................+++
.......+++
e is 65537 (0x10001)
```

In order to generate the CA certificate:

```
[jedepuyd@KJK-SRVIOT-10 signing]$ openssl req -x509 -new -nodes -key rootca-key.pem -sha256 -
days 4096 -out rootca-cert.pem
You are about to be asked to enter information that is incorporated
```

```
into your certificate request.
What you are about to enter is what is called a Distinguished Name (DN).
There are quite a few fields but you can leave some blank
For some fields there can be a default value,
If you enter '.', the field can be left blank.
-----
Country Name (2 letter code) [XX]:BE
State or Province Name (full name) []:WVL
Locality Name (eg, city) [Default City]:Kortrijk
Organization Name (eg, company) [Default Company Ltd]:Cisco
Organizational Unit Name (eg, section) []:IOT
Common Name (eg, your name or your server's hostname) []:ioxrootca
Email Address []:
```

The values in the CA certificate must be adjusted to match your use case.

Step 2. Generate Trust Anchor for Use on IOx

Now that you have the necessary key and certificate for your CA, you can create a trust anchor bundle for use on your IOx device. The trust anchor bundle must contain the full CA signing chain (in case intermediate certificate are used for signing) and a info.txt file which is used to provide the (free form) metadata.

First, create the info.txt file and put some metadata in it:

```
[jedepuyd@KJK-SRVIOT-10 signing]$ echo "iox app root ca v1">info.txt
```

Optionally, if you have multiple CA certificates, to form your CA certificate chain, you need to put them together in one .pem:

```
cat first_cert.pem second_cert.pem > combined_cert.pem
```

> **Note**: This step is not required for this article, since a single CA root certificate is used to direct sign, this is not recommended for production and the root CA keypair must always be stored offline.

The CA certificate chain needs to be named ca-chain.cert.pem, so prepare this file:

```
[jedepuyd@KJK-SRVIOT-10 signing]$ cp rootca-cert.pem ca-chain.cert.pem
```

Finally, you can combine the ca-chain.cert.pem and info.txt in a gzipped tar:

```
[jedepuyd@KJK-SRVIOT-10 signing]$ tar -czf trustanchorv1.tar.gz ca-chain.cert.pem info.txt
```

Step 3. Import Trust Anchor on IOx-Device

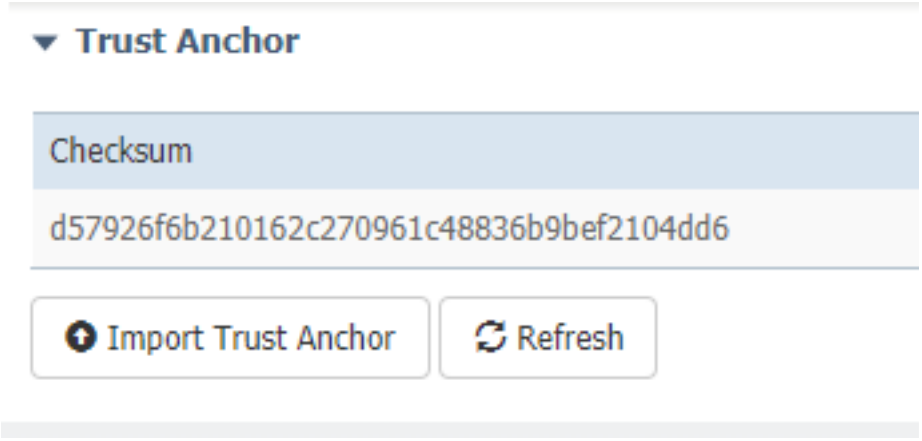
The trustanchorv1.tar.gz which you created in the previous step needs to be imported onto your IOx-device. The files in the bundle are used to verify if an application got signed with a CA-signed certificate from the correct CA before it allows an installation.

The import of the trust anchor can be done via the ioxlcient:

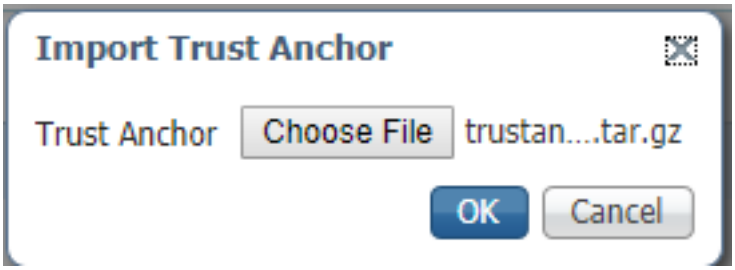
```
[jedepuyd@KJK-SRVIOT-10 signing]$ ioxclient platform signedpackages trustanchor set
trustanchorv1.tar.gz
Currently active profile :  default
Command Name:  plt-sign-pkg-ta-set
Response from the server:  Imported trust anchor file successfully
[jedepuyd@KJK-SRVIOT-10 signing]$ ioxclient platform signedpackages enable
Currently active profile :  default
Command Name:  plt-sign-pkg-enable
Successfully updated the signed package deployment capability on the device to true
```

Another option is to import the trust anchor via Local Manager:

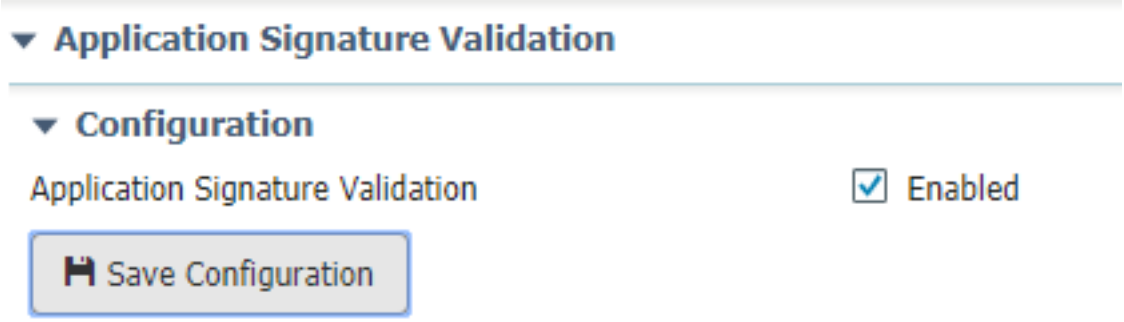
Navigate to **System Setting > Import Trust Anchor** as shown in the image.



Select the file which you generated in Step 2. and click **OK** as shown in the image.



After you have successfully imported the trust anchor, check **Enabled** for **Application Signing Validation** and click **Save Configuration** as shown in the image:



Step 4. Create Application Specific Key and CSR

Next, you can create a key and certificate pair that is used to sign into your IOx application. The best practice is to generate one specific keypair for each application you plan to deploy.

As long as each of those is signed with the same CA, they are all considered as valid.

In order to generate the application specific key:

```
[jedepuyd@KJK-SRVIOT-10 signing]$ openssl genrsa -out app-key.pem 2048
Generating RSA private key, 2048 bit long modulus
.......+++
...+++
e is 65537 (0x10001)
```

In order to generate the CSR:

```
[jedepuyd@KJK-SRVIOT-10 signing]$ openssl req -new -key app-key.pem -out app.csr
You are about to be asked to enter information that is incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name (DN).
There are quite a few fields but you can leave some blank.
For some fields there can be a default value,
If you enter '.', the field can be left blank.
-----
Country Name (2 letter code) [XX]:BE
State or Province Name (full name) []:WVL
Locality Name (eg, city) [Default City]:Kortrijk
Organization Name (eg, company) [Default Company Ltd]:Cisco
Organizational Unit Name (eg, section) []:IOT
Common Name (eg, your name or your server's hostname) []:ioxapp
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

As with the CA, the values in the application certificate must be adjusted to match your use case.

Step 5. Sign Application Specific Certificate with CA

Now that you have the requirements for your CA and application CSR, you can sign the CSR with the use of CA. The result is a signed application-specific certificate:

```
[jedepuyd@KJK-SRVIOT-10 signing]$ openssl x509 -req -in app.csr -CA rootca-cert.pem -CAkey
rootca-key.pem -CAcreateserial -out app-cert.pem -days 4096 -sha256
Signature ok
subject=/C=BE/ST=WVL/L=Kortrijk/O=Cisco/OU=IOT/CN=ioxapp
Getting CA Private Key
```

Step 6. Package your IOx Application and Sign it with Application Specific Certificate

At this point, you are ready to package your IOx application and to sign it with the generated keypair from Step 4. and signed by the CA in Step 5.

The rest of the process to create the source and package.yaml for your application remains

unchanged.

package IOx application with the use of keypair:

```
[jedepuyd@KJK-SRVIOT-10 iox_docker_pythonsleep]$ ioxclient package --rsa-key ../signing/app-
key.pem --certificate ../signing/app-cert.pem .
Currently active profile :  default
Command Name:  package
Using rsa key and cert provided via command line to sign the package
Checking if package descriptor file is present..
Validating descriptor file /home/jedepuyd/iox/iox_docker_pythonsleep/package.yaml with package
schema definitions
Parsing descriptor file..
Found schema version  2.2
Loading schema file for version  2.2
Validating package descriptor file..
File /home/jedepuyd/iox/iox_docker_pythonsleep/package.yaml is valid under schema version 2.2
Created Staging directory at :  /tmp/666018803
Copying contents to staging directory
Checking for application runtime type
Couldn't detect application runtime type
Creating an inner envelope for application artifacts
Excluding  .DS_Store
Generated  /tmp/666018803/artifacts.tar.gz
Calculating SHA1 checksum for package contents..
Package MetaData file was not found at  /tmp/666018803/.package.metadata
Wrote package metadata file :  /tmp/666018803/.package.metadata
Root Directory :  /tmp/666018803
Output file:  /tmp/096960694
Path:  .package.metadata
SHA1 : 2a64461a921c2d5e8f45e92fe203127cf8a06146
Path:  artifacts.tar.gz
SHA1 : 63da3eb3d81e13249b799bf57845f3fc9f6f2f94
Path:  package.yaml
SHA1 : 0e6259e49ff22d6d38e6d1913759c5674c5cec6d
Generated package manifest at  package.mf
Signed the package and the signature is available at package.cert
Generating IOx Package..
Package generated at /home/jedepuyd/iox/iox_docker_pythonsleep/package.tar
```

Step 7. Deploy your Signed IOx Package onto a Signature-Enabled Device

Last step in the process would be to deploy the application to your IOx device. There is no difference in comparison to an unsigned application deployment:

```
[jedepuyd@KJK-SRVIOT-10 iox_docker_pythonsleep]$ ioxclient app install test package.tar
Currently active profile :  default
Command Name:  application-install
Saving current configuration
Installation Successful. App is available at :
https://10.50.215.248:8443/iox/api/v2/hosting/apps/test
Successfully deployed
```

Verify

Use this section in order to confirm that your configuration works properly.

In order to verify if an application key is correctly signed with your CA, you can do this:

```
[jedepuyd@KJK-SRVIOT-10 signing]$ openssl verify -CAfile rootca-cert.pem app-cert.pem
app-cert.pem: OK
```

Troubleshoot

This section provides information you can use in order to troubleshoot your configuration.

When you experience issues with the deployment of applications, you could see one of these errors:

```
[jedepuyd@KJK-SRVIOT-10 iox_docker_pythonsleep]$ ioxclient app install test package.tar
Currently active profile :  default
Command Name:  application-install
Saving current configuration
Could not complete your command :  Error. Server returned 500
{
 "description": "Invalid Archive file: Certificate verification failed: [18, 0, 'self signed
certificate']",
 "errorcode": -1,
 "message": "Invalid Archive file"
}
```

Something went wrong in signing the application certificate with the use of the CA or it doesn't match with the one in the trusted anchor bundle.

Use the instructions mentioned in Verify section, to check your certificates and also the trusted anchor bundle as well.

These error indicates that your package was not signed correctly, you can look into Step 6. again.

```
[jedepuyd@KJK-SRVIOT-10 iox_docker_pythonsleep]$ ioxclient app install test2 package.tar
Currently active profile :  default
Command Name:  application-install
Saving current configuration
Could not complete your command :  Error. Server returned 500
{
 "description": "Package signature file package.cert or package.sign not found in package",
 "errorcode": -1009,
 "message": "Error during app installation"
}
```