Configurez la validation de signature de module d'IOx

Contenu

Introduction

Ce document décrit d'une manière détaillée comment créer et utiliser les modules signés sur la plate-forme d'IOx.

Conditions préalables

Conditions requises

Cisco vous recommande de prendre connaissance des rubriques suivantes :

- La connaissance de base de Linux
- Comprenez comment les Certificats fonctionnent

[Composants utilisés](#)

Les informations contenues dans ce document sont basées sur les versions de matériel et de logiciel suivantes :

- Périphérique capable d'IOx qui est configuré pour IOx :
  Adresse IP configuréeSystème d'exploitation client (gos) et cadre d'applications de Cisco (CAF) ces passagesTraduction d'adresses de réseau (NAT) configuré pour l'accès à CAF

(port 8443)
- Hôte de Linux avec Secure Sockets Layer ouvert (SSL) installé
- Fichiers d'installation de client d'IOx dont peut être téléchargé : https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

Les informations contenues dans ce document ont été créées à partir des périphériques d'un environnement de laboratoire spécifique. Tous les périphériques utilisés dans ce document ont démarré avec une configuration effacée (par défaut). Si votre réseau est opérationnel, assurez-vous que vous comprenez l'effet potentiel de toute commande.

Informations générales

Depuis la release d'IOx, la signature de progiciel AC5 est prise en charge. Cette caractéristique laisse s'assurer que le progiciel est valide et celui installé sur le périphérique est obtenu d'une source sûre. Si la validation de signature de progiciel est activée dans une plate-forme, seulement des applications alors signées peuvent être déployées.

Configurez

Ces étapes sont exigées pour utiliser la validation de signature de module :

1. Créez une clé d'Autorité de certification (CA) et la délivrez un certificat.
2. Générez une ancre de confiance pour l'usage sur IOx.
3. Importez l'ancre de confiance sur votre IOx-périphérique.
4. Créez une demande de signature spécifique à l'application de clé et de certificat (CSR).
5. Signez le certificat spécifique à l'application avec l'utilisation du CA.
6. Empaquetez votre application d'IOx, signez-la avec le certificat spécifique à l'application.
7. Déployez votre module signé d'IOx sur un périphérique signature-activé.

Note: Pour cet article, un CA auto-signé est utilisé dans un scénario de production. La meilleure option est d'employer un fonctionnaire CA ou le CA de votre société pour signer.

Note: Les options pour le CA, les clés et les signatures sont choisies pour le laboratoire seulement et pourraient devoir être ajustées pour votre environnement.

Étape 1. Créez la clé CA et la délivrez un certificat

La première étape est de créer votre propre CA. Ceci peut être simplement fait par la génération d'une clé pour le CA et d'un certificat pour cette clé :

Afin de générer la clé CA :

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

Afin de générer le certificat de CA :

```
[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 []:
```

Les valeurs dans le certificat de CA doivent être ajustées pour apparier votre cas d'utilisation.

Étape 2. Générez l'ancre de confiance pour l'usage sur IOx

Maintenant que vous avez la clé et le certificat nécessaires pour votre CA, vous pouvez créer un paquet d'ancre de confiance pour l'usage sur votre périphérique d'IOx. Le paquet d'ancre de confiance doit contenir la pleine chaîne de signature CA (au cas où le certificat intermédiaire seraient utilisés pour la signature) et un fichier d'info.txt qui est utilisé pour fournir les métadonnées (de forme libre).

D'abord, créez le fichier d'info.txt et mettez quelques métadonnées dans lui :

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

Sur option, si vous avez de plusieurs Certificats CA, pour former votre chaîne de certificat de CA, vous devez les remonter dans un .pem :

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

> **Note**: Cette étape n'est pas exigée pour cet article, puisqu'un certificat racine simple CA est utilisé pour diriger le signe, ceci n'est pas recommandée pour la production et le keypair de la racine CA doit toujours être enregistré off-line.

La chaîne de certificat de CA doit être nommée ca-chain.cert.pem, ainsi préparez ce fichier :

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

En conclusion, vous pouvez combiner le ca-chain.cert.pem et info.txt dans un goudron gzipped :

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

Étape 3. Ancre de confiance d'importation sur l'IOx-périphérique

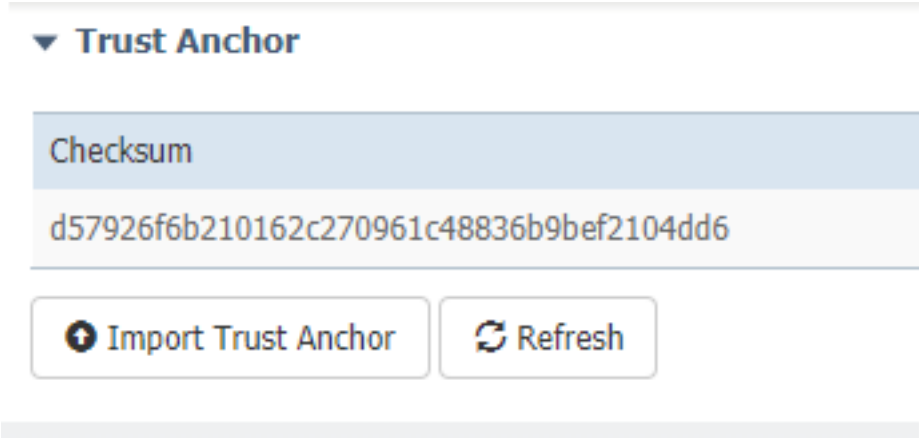
trustanchorv1.tar.gz que vous avez créé dans l'étape précédente doit être importé sur votre IOx-périphérique. Les fichiers dans le paquet sont utilisés pour vérifier si une application obtenue signait avec un certificat Ca-signé du CA correct avant qu'elle permette une installation.

L'importation de l'ancre de confiance peut être faite par l'intermédiaire de l'ioxlcient :

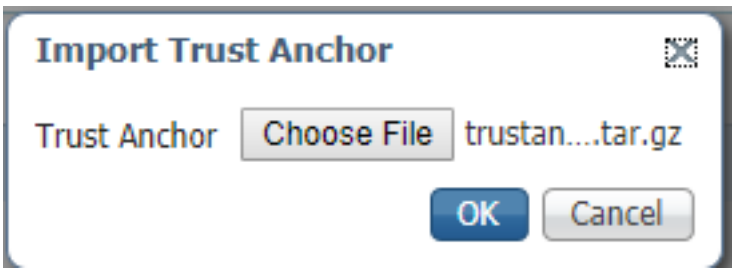
```
[jedepuyd@KJK-SRVIOT-10 signing]$ ioxclient platform signedpackages trustanchor set
trustanchorv1.tar.gz
Currently active profile :  default
Command Name:  plt-sign-pkg-ta-set
Response from the server:  Imported trust anchor file successfully
[jedepuyd@KJK-SRVIOT-10 signing]$ ioxclient platform signedpackages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

Une autre option est d'importer l'ancre de confiance par l'intermédiaire du gestionnaire local :

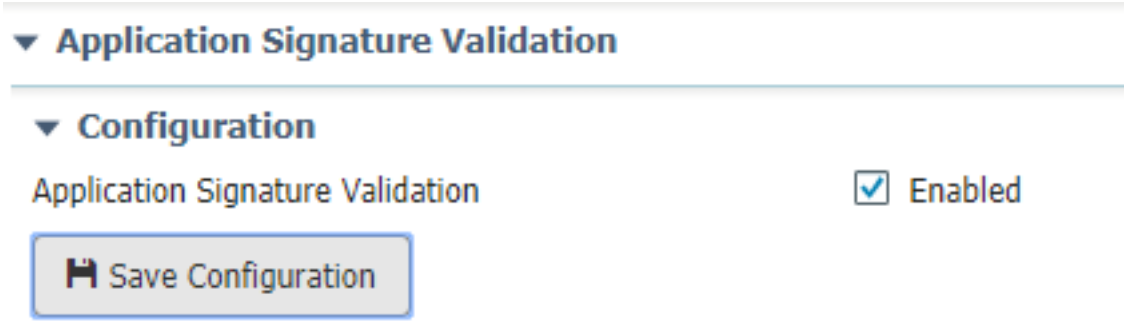
Naviguez vers l'**ancre de confiance de paramètre système > d'importation** suivant les indications de l'image.



Sélectionnez le fichier que vous avez généré dans l'étape 2. et cliquez sur OK suivant les indications de l'image.



Après que vous ayez avec succès importé l'ancre de confiance, le contrôle **activé** pour la **validation de signature d'application** et la **save configuration de** clic suivant les indications de l'image :



Étape 4. Créez la clé spécifique à l'application et le CSR

Ensuite, vous pouvez créer une clé et délivrer un certificat la paire qui est utilisée pour signer dans votre application d'IOx. La pratique recommandée est de générer un keypair spécifique pour chaque application que vous prévoyez de se déployer.

Tant que chacune de ceux est signée avec le même CA, ils tous sont considérés en tant que valide.

Afin de générer la clé spécifique à l'application :

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

Afin de générer le 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 []:
```

Comme avec le CA, les valeurs dans le certificat d'application doivent être ajustées pour apparier votre cas d'utilisation.

Étape 5. Certificat spécifique à l'application de signe avec le CA

Maintenant que vous avez les conditions requises pour votre CA et CSR d'application, vous pouvez signer le CSR avec l'utilisation du CA. Le résultat est un certificat spécifique à l'application signé :

```
[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
```

Étape 6. Empaquetez votre application d'IOx et signez-la avec le certificat spécifique à l'application

En ce moment, vous êtes prêt à empaqueter votre application d'IOx et à la signer avec le keypair généré de l'étape 4. et avez signé par le CA dans l'étape 5.

Le reste du processus pour créer la source et le package.yaml pour votre application demeure sans changement.

empaquetez l'application d'IOx avec l'utilisation du 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
```

Étape 7. Déployez votre module signé d'IOx sur un périphérique Signature-activé

La dernière étape dans le processus serait de déployer l'application vers votre périphérique d'IOx. Il n'y a aucune différence par rapport à un déploiement d'applications non signé :

```
[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
```

Vérifiez

Utilisez cette section pour confirmer que votre configuration fonctionne correctement.

Afin de vérifier si une clé d'application est correctement signée avec votre CA, vous pouvez faire ceci :

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

Dépannez

Cette section fournit des informations que vous pouvez utiliser pour dépanner votre configuration.

Quand vous éprouvez des questions avec le déploiement des applications, vous pourriez voir une de ces erreurs :

```
[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"
}
```

Quelque chose est allée mal en signant le certificat d'application avec l'utilisation du CA ou elle ne s'assortit pas avec celui dans le paquet de confiance d'ancre.

Utilisez les instructions mentionnées dedans vérifient la section, pour vérifier vos Certificats et également le paquet de confiance d'ancre aussi bien.

Ces l'erreur indique que votre module n'a pas été signé correctement, vous peut examiner l'étape 6. de nouveau.

```
[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"
}
```