IOxパッケージシグニチャ検証の設定

内容

概要

このドキュメントでは、IOxプラットフォームで署名付きパッケージを作成および使用する方法について詳しく説明します。

前提条件

要件

次の項目に関する知識があることが推奨されます。

- Linux に関する基本知識
- 証明書の仕組みを理解する

使用するコンポーネント

このドキュメントの情報は、次のソフトウェアとハードウェアのバージョンに基づいています。

- IOx対応のデバイス。IOx用に設定されます。
  設定されたIPアドレス実行するゲストオペレーティングシステム(GOS)およびCisco Application Framework(CAF)CAF（ポート8443）へのアクセス用に設定されたネットワークアドレス変換(NAT)
- オープンSSL(Secure Sockets Layer)がインストールされたLinuxホスト
- IOxクライアントのインストールファイル（ダウンロード可能）:https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=2863067

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、初期（デフォルト）設定の状態から起動しています。対象のネットワークが実稼働中である場合には、どのようなコマンドについても、その潜在的な影響について確実に理解しておく必要があります。

背景説明

IOxリリース以降、AC5アプリケーションパッケージ署名がサポートされています。この機能を使用すると、アプリケーションパッケージが有効で、デバイスにインストールされているパッケージが信頼できるソースから取得されていることを確認できます。プラットフォームでアプリケーションパッケージシグニチャ検証がオンになっている場合は、署名されたアプリケーションのみを展開できます。

設定

パッケージの署名検証を使用するには、次の手順が必要です。

1. 認証局(CA)キーと証明書を作成します。
2. IOxで使用する信頼アンカーを生成します。
3. IOxデバイスに信頼アンカーをインポートします。
4. アプリケーション固有のキーと証明書署名要求(CSR)を作成します。
5. CAを使用して、アプリケーション固有の証明書に署名します。
6. IOxアプリケーションをパッケージ化し、アプリケーション固有の証明書で署名します。
7. 署名可能なデバイスに署名付きIOxパッケージを導入します。

注：この記事では、自己署名CAを実稼働シナリオで使用します。最適なオプションは、公式CAまたは会社のCAを使用して署名することです。

注：CA、キー、およびシグニチャのオプションはラボ目的でのみ選択され、環境に合わせて調整する必要がある場合があります。

ステップ1:CAキーと証明書の作成

最初の手順は、独自のCAを作成することです。これは、CAのキーとそのキーの証明書を生成するだけで実行できます。

CAキーを生成するには：

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

CA証明書を生成するには：

```
[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 []:
```

CA証明書の値は、ユースケースに合わせて調整する必要があります。

ステップ2:IOxで使用する信頼アンカーの生成

これで、CAに必要なキーと証明書が揃ったので、IOxデバイスで使用する信頼アンカーバンドルを作成できます。信頼アンカーバンドルには、完全なCA署名チェーン（中間証明書が署名に使用される場合）と（自由形式）メタデータの提供に使用されるinfo.txtファイルが含まれている必要があります。

まず、info.txtファイルを作成し、メタデータを次のように追加します。

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

必要に応じて、複数のCA証明書がある場合、CA証明書チェーンを形成するには、それらを1つの.pemにまとめる必要があります。

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

> **注**：この記事では、1つのCAルート証明書を直接署名に使用するため、この手順は不要です。これは実稼働環境では推奨されず、ルートCAキーペアは常にオフラインで保存する必要があります。

CA証明書チェーンはca-chain.cert.pemという名前にする必要があるため、次のファイルを準備します。

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

最後に、ca-chain.cert.pemとinfo.txtをgzip tar形式で結合できます。

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

ステップ3:IOxデバイスでの信頼アンカーのインポート

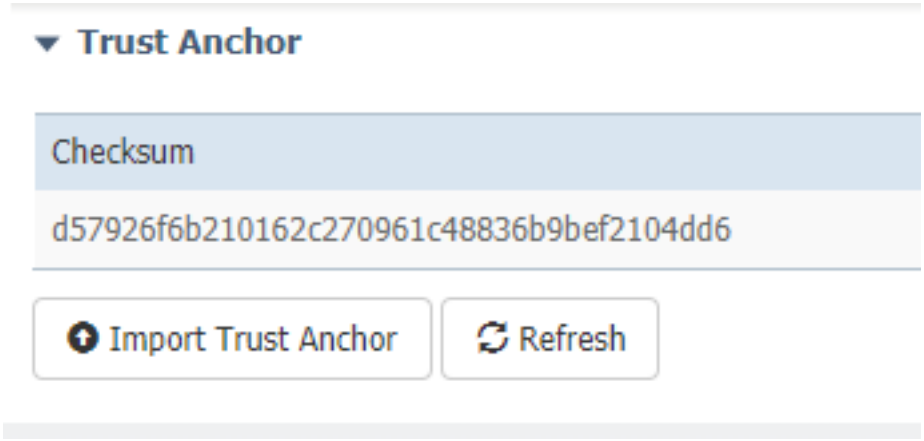
前の手順で作成したtrustanchorv1.tar.gzをIOxデバイスにインポートする必要があります。バンドル内のファイルは、インストールを許可する前に、アプリケーションが正しいCAからCA署名付き証明書で署名されたかどうかを確認するために使用されます。

信頼アンカーのインポートは、次のキーワードを使用して実行できます。

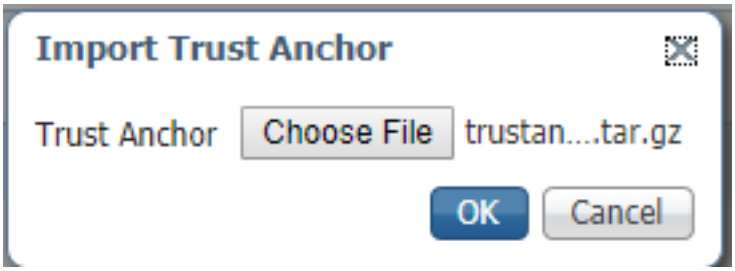
```
[jedepuyd@KJK-SRVIOT-10 signing]$ ioxclient platform signedpackages trustanchor set
trustanchorv1.tar.gz
Currently active profile :  default
Command Name:  plt-sign-pkg-ta-set
Response from the server:  Imported trust anchor file successfully
[jedepuyd@KJK-SRVIOT-10 signing]$ ioxclient platform signedpackages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

もう1つのオプションは、ローカルマネージャを介して信頼アンカーをインポートすることです。

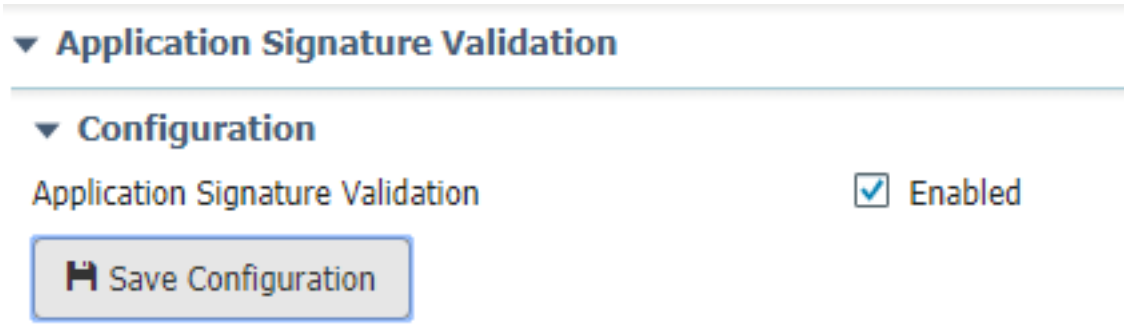
図に示すように、**[System Setting] > [Import Trust Anchor]**に移動します。



ステップ2.で生成したファイルを選択し、図に示すように**[OK]**をクリックします。



信頼アンカーを正常にインポートした後、次の図に示すように、**[Application Signing Validation]**の[Enabled]をオンにし、**[Save Configuration]**をクリックします。



ステップ4：アプリケーション固有のキーとCSRの作成

次に、IOxアプリケーションへのサインインに使用されるキーと証明書のペアを作成できます。ベストプラクティスは、展開するアプリケーションごとに1つの特定のキーペアを生成することです。

それぞれのCAが同じCAで署名されている限り、それらはすべて有効と見なされます。

アプリケーション固有のキーを生成するには、次の手順を実行します。

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

CSRを生成するには：

```
[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 []:
```

CAと同様に、アプリケーション証明書の値はユースケースに合わせて調整する必要があります。

ステップ5:CAを使用したアプリケーション固有証明書の署名

CAおよびアプリケーションCSRの要件が整ったので、CAを使用してCSRに署名できます。その結果、署名されたアプリケーション固有の証明書が作成されます。

```
[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
```

ステップ6:IOxアプリケーションをパッケージ化し、アプリケーション固有証明書で署名する

これで、IOxアプリケーションをパッケージ化し、ステップ4.から生成されたキーペアで署名し、

ステップ5でCAによって署名する準備ができました。

アプリケーションのソースとpackage.yamlを作成する残りのプロセスは変更されません。

キーペアを使用したIOxアプリケーションのパッケージ：

```
[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
```

ステップ7：署名可能なデバイスへの署名付きIOxパッケージの展開

このプロセスの最後のステップは、IOxデバイスにアプリケーションを導入することです。署名されていないアプリケーションの導入と比較しても違いはありません。

```
[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
```

確認

ここでは、設定が正常に機能しているかどうかを確認します。

アプリケーションキーがCAで正しく署名されているかどうかを確認するには、次の手順を実行します。

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

トラブルシュート

ここでは、設定のトラブルシューティングに使用できる情報を示します。

アプリケーションの導入に関する問題が発生すると、次のいずれかの工ラーが表示されます。

```
[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"
}
```

CAを使用してアプリケーション証明書に署名するときに問題が発生したか、信頼されたアンカーバンドル内のアプリケーション証明書と一致しません。

「確認」セクションで説明されている手順を使用して、証明書および信頼できるアンカーバンドルも確認します。

これらのエラーは、パッケージが正しく署名されていないことを示します。手順6を再度確認できます。

```
[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"
}
```