

IOx 패키지 서명 유효성 검사 구성

목차

[소개](#)

[사전 요구 사항](#)

[요구 사항](#)

[사용되는 구성 요소](#)

[배경 정보](#)

[구성](#)

[1단계. CA 키 및 인증서 생성](#)

[2단계. IOx에서 사용할 트러스트 앵커 생성](#)

[3단계. IOx-Device에 Trust Anchor 가져오기](#)

[4단계. 애플리케이션별 키 및 CSR 생성](#)

[5단계. CA로 애플리케이션별 인증서 서명](#)

[6단계. IOx 애플리케이션을 패키징하고 애플리케이션별 인증서로 서명](#)

[7단계. 서명된 IOx 패키지를 시그니처가 활성화된 디바이스에 구축합니다.](#)

[다음을 확인합니다.](#)

[문제 해결](#)

소개

이 문서에서는 IOx 플랫폼에서 서명된 패키지를 생성하고 사용하는 방법에 대해 자세히 설명합니다.

사전 요구 사항

요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- 기본 Linux 지식
- 인증서 작동 방식 이해

사용되는 구성 요소

이 문서의 정보는 다음 소프트웨어 및 하드웨어 버전을 기반으로 합니다.

- IOx용으로 구성된 IOx 지원 디바이스:
 - 구성된 IP 주소
 - GOS(Guest Operating System) 및 Cisco CAF(Application Framework) 실행
 - CAF(포트 8443)에 액세스하도록 구성된 NAT(Network Address Translation)
- 개방형 SSL(Secure Sockets Layer)이 설치된 Linux 호스트

- 다운로드 가능한 IOx 클라이언트 설치 파일:

<https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762>

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우, 모든 명령어의 잠재적인 영향을 미리 숙지하시기 바랍니다.

배경 정보

IOx 릴리스 이후 AC5 애플리케이션 패키지 서명이 지원됩니다. 이 기능을 사용하면 응용 프로그램 패키지가 유효하고 장치에 설치된 패키지를 신뢰할 수 있는 소스에서 가져올 수 있습니다. 플랫폼에서 응용 프로그램 패키지 서명 유효성 검사가 켜져 있으면 서명된 응용 프로그램만 배포할 수 있습니다.

구성

패키지 서명 검증을 사용하려면 다음 단계가 필요합니다.

1. CA(Certificate Authority) 키 및 인증서를 생성합니다.
2. IOx에서 사용할 트러스트 앵커를 생성합니다.
3. IOx 디바이스에서 트러스트 앵커를 가져옵니다.
4. 애플리케이션별 키 및 CSR(Certificate Signing Request)을 생성합니다.
5. CA를 사용하여 애플리케이션별 인증서에 서명합니다.
6. IOx 애플리케이션을 패키지화하고 애플리케이션별 인증서로 서명합니다.
7. 서명된 IOx 패키지를 서명 지원 디바이스에 구축합니다.

 참고: 이 문서에서는 프로덕션 시나리오에서 자체 서명 CA를 사용합니다. 가장 좋은 방법은 공식 CA 또는 회사의 CA를 사용하여 서명하는 것입니다.

 참고: CA, 키 및 서명 옵션은 랩 목적으로만 선택되며 환경에 맞게 조정해야 할 수 있습니다.

1단계. CA 키 및 인증서 생성

첫 번째 단계는 고유한 CA를 생성하는 것입니다. 이는 CA에 대한 키 및 해당 키에 대한 인증서를 생성하면 간단하게 수행할 수 있습니다.

CA 키를 생성하려면

```
[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 인증서를 생성하려면

```
[jedepuyd@KJK-SRVIOT-10 signing]$ openssl req -x509 -new -nodes -key rootca-key.pem -sha256 -days 4096
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 인증서의 값은 사용 사례와 일치하도록 조정해야 합니다.

2단계. IOx에서 사용할 트러스트 앵커 생성

이제 CA에 필요한 키와 인증서가 있으므로 IOx 디바이스에서 사용할 트러스트 앵커 번들을 생성할 수 있습니다. 트러스트 앵커 번들에는 전체 CA 서명 체인(서명에 중간 인증서가 사용되는 경우)과 (자유 형식) 메타데이터를 제공하는 데 사용되는 info.txt 파일이 포함되어 있어야 합니다.

먼저 info.txt 파일을 만들고 몇 가지 메타데이터를 넣습니다.

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

선택적으로, 여러 CA 인증서가 있는 경우 CA 인증서 체인을 구성하려면 .pem 하나로 통합해야 합니다.

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

 참고: 단일 CA 루트 인증서가 직접 서명에 사용되므로 이 단계는 이 문서에 필요하지 않습니다. 이는 프로덕션에 권장되지 않으며 루트 CA 키 쌍은 항상 오프라인으로 저장해야 합니다.

CA 인증서 체인의 이름은 ca-chain.cert.pem이어야 하므로 다음 파일을 준비합니다.

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

마지막으로, gzipped tar에서 ca-chain.cert.pem 및 info.txt를 결합할 수 있습니다.

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

3단계. IOx-Device에 Trust Anchor 가져오기

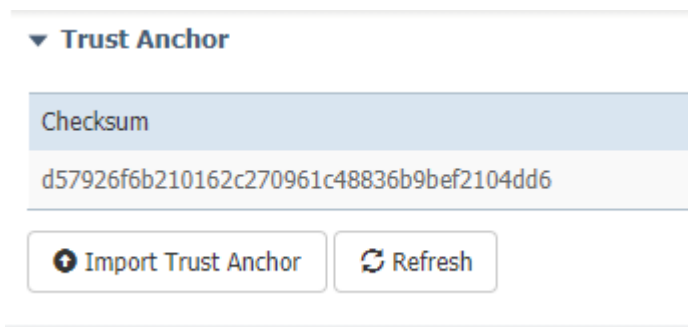
이전 단계에서 생성한 trustanchorv1.tar.gz를 IOx 디바이스로 가져와야 합니다. 번들의 파일은 애플리케이션이 설치를 허용하기 전에 올바른 CA에서 CA 서명 인증서를 사용하여 서명되었는지 확인하는 데 사용됩니다.

Trust Anchor 가져오기는 ioxlent를 통해 수행할 수 있습니다.

```
[jedepuyd@KJK-SRVIOT-10 signing]$ ioxclient platform signedpackages trustanchor set trustanchorv1.tar.g
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
```

또 다른 옵션은 로컬 관리자를 통해 트러스트 앵커를 가져오는 것입니다.

이미지에 표시된 대로 System Setting(시스템 설정) > Import Trust Anchor(Trust Anchor 가져오기)로 이동합니다.

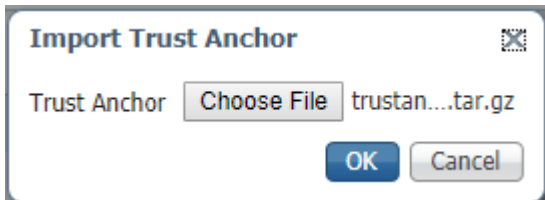


▼ Trust Anchor

Checksum
d57926f6b210162c270961c48836b9bef2104dd6

ⓘ Import Trust Anchor ↻ Refresh

2단계에서 생성한 파일을 선택하고 그림과 같이 OK를 클릭합니다.



트러스트 앵커를 성공적으로 가져온 후 이미지에 표시된 대로 Enabled for Application Signing Validation(애플리케이션 서명 검증에 대해 활성화됨)을 선택하고 Save Configuration(컨피그레이션 저장)을 클릭합니다.

▼ Application Signature Validation

▼ Configuration

Application Signature Validation ☒ Enabled

 Save Configuration

4단계. 애플리케이션별 키 및 CSR 생성

그런 다음 IOx 애플리케이션에 로그인하는 데 사용되는 키 및 인증서 쌍을 생성할 수 있습니다. 모범 사례는 구축하려는 각 애플리케이션에 대해 하나의 특정 키 쌍을 생성하는 것입니다.

각 CA가 동일한 CA로 서명된 경우 모두 유효한 것으로 간주됩니다.

애플리케이션별 키를 생성하려면

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

CSR을 생성하려면

```
[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 []:

CA와 마찬가지로, 애플리케이션 인증서의 값을 사용 사례와 일치하도록 조정해야 합니다.

5단계. CA로 애플리케이션별 인증서 서명

이제 CA 및 애플리케이션 CSR에 대한 요구 사항이 있으므로 CA를 사용하여 CSR에 서명할 수 있습니다. 서명된 애플리케이션별 인증서는 다음과 같습니다.

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

6단계. IOx 애플리케이션을 패키징하고 애플리케이션별 인증서로 서명

이제 IOx 애플리케이션을 패키징하고 4단계에서 생성된 키 쌍으로 서명하고 5단계에서 CA가 서명할 준비가 되었습니다.

응용 프로그램의 source 및 package.yaml을 만드는 나머지 프로세스는 변경되지 않습니다.

keypair를 사용하여 IOx 애플리케이션 패키지:

```
[jedepuyd@KJK-SRVIOT-10 iox_docker_pythonsleep]$ ioxclient package --rsa-key ../signing/app-key.pem --c  
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 d  
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
```

7단계. 서명된 IOx 패키지를 시그니처가 활성화된 디바이스에 구축합니다.

이 프로세스의 마지막 단계는 애플리케이션을 IOx 디바이스에 구축하는 것입니다. 서명되지 않은 애플리케이션 배포와 비교해도 차이가 없습니다.

```
[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
```

다음을 확인합니다.

구성이 올바르게 작동하는지 확인하려면 이 섹션을 활용하십시오.

애플리케이션 키가 CA에 올바르게 서명되었는지 확인하려면 다음을 수행할 수 있습니다.

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

문제 해결

이 섹션에서는 설정 문제 해결을 위해 사용할 수 있는 정보를 제공합니다.

애플리케이션 구축에 문제가 발생하면 다음 오류 중 하나가 표시될 수 있습니다.

```
[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 certificat
  "errorcode": -1,
  "message": "Invalid Archive file"
}
```

CA를 사용하여 응용 프로그램 인증서를 서명하는 동안 문제가 발생했습니다. 그렇지 않으면 신뢰할 수 있는 앵커 번들의 인증서와 일치하지 않습니다.

Verify(확인) 섹션에 설명된 지침을 사용하여 인증서 및 신뢰할 수 있는 앵커 번들도 확인합니다.

이 오류는 패키지가 올바르게 서명되지 않았음을 나타냅니다. 6단계를 다시 살펴볼 수 있습니다.

```
[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"
}
```

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.