配置IOx包签名验证

目录

简介

本文描述用一个详细的方式如何创建和使用在IOx平台的签字的包。

先决条件

要求

Cisco 建议您了解以下主题：

- 基本Linux知识
- 知道证书如何工作

使用的组件

本文档中的信息基于以下软件和硬件版本：

- 为IOx配置的IOx有能力设备：
  配置的IP地址访客操作系统(GOS)和Cisco应用框架(CAF)该运行为对CAF (端口8443)的访问配置的网络地址转换(NAT)
- 有安装的开放安全套接字协议层(SSL)的Linux主机
- IOx能下载的客户端安装文件：https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原

始（默认）配置。如果您使用的是真实网络，请确保您已经了解所有命令的潜在影响。

背景信息

从IOx版本， AC5支持应用程序签字。此功能准许保证应用程序有效，并且在设备安装的那个从可信的源获取。如果应用程序签名验证在平台打开，那时签字的应用程序可以被实施。

配置

这些步骤要求使用包签名验证：

1. 创建Certificate Authority (CA)密钥和证书。
2. 生成一个信任锚点为在IOx的使用。
3. 导入在您的IOX设备的信任锚点。
4. 创建一有特殊用途的密钥和证书签名请求(CSR)。
5. 签署与使用的有特殊用途的证书CA。
6. 包您的IOx应用程序，签署它与有特殊用途的证书。
7. 部署您的在一个签名启用的设备上的签字的IOx包。

Note:对于此条款，自已签署的CA用于制作方案。最好的选项是使用正式CA或您的公司的CA签字。

Note:CA、密钥和签名的选项选择只为实验室目的并且也许需要为您的环境调节。

步骤1.创建CA密钥和证书

第一步将创建您自己的CA。这可以由一密钥和该密钥的一证书的生成CA的完成：

为了生成CA密钥：

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

为了生成CA证书：

```
[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 []:
```

必须调节在CA证书的值匹配您的用例。

步骤2.生成托拉斯锚点为在IOx的使用

即然您有必要的密钥和证书您的CA的，您能创建一个信任锚点套件为使用在您的IOx设备。信任锚点套件必须包含签署一系列的全双工CA (万一中间证书使用签字)和用于提供的info.txt文件(自由格式)元数据。

首先，请创建info.txt文件并且放置某元数据在它：

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

随意地，如果有多个CA证书，形成您的CA证书一系列，您在一.pem需要汇集他们：

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

> **Note**:此步骤没有为此条款要求，因为单个CA根证明用于处理符号，这没有为制作推荐，并且根CA密钥对一定总是存储的脱机。

CA证书一系列需要被命名CAchain.cert.pem，因此请准备此文件：

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

最后，您能结合CAchain.cert.pem和info.txt在gzipped tar ：

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

步骤3.导入在IOX设备的托拉斯锚点

您在上一步创建的trustanchorv1.tar.gz需要导入在您的IOX设备上。在套件的文件用于验证，如果获得的应用程序签了字与从正确CA的一个CA签发的证书，在允许安装前。

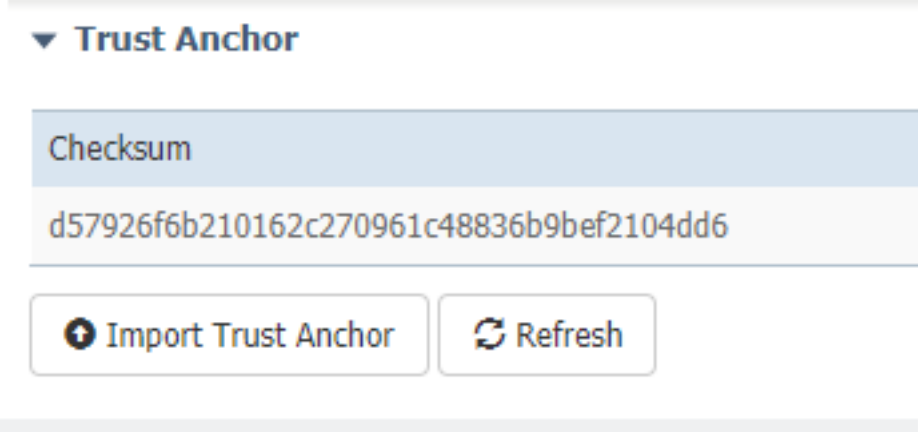
信任锚点的导入可以通过ioxlcient完成：

```
[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
```

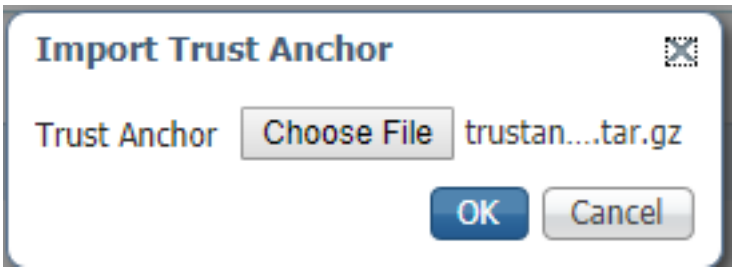
```
Command Name:  plt-sign-pkg-enable
Successfully updated the signed package deployment capability on the device to true
```

另一个选项是导入信任锚点通过当地干事：

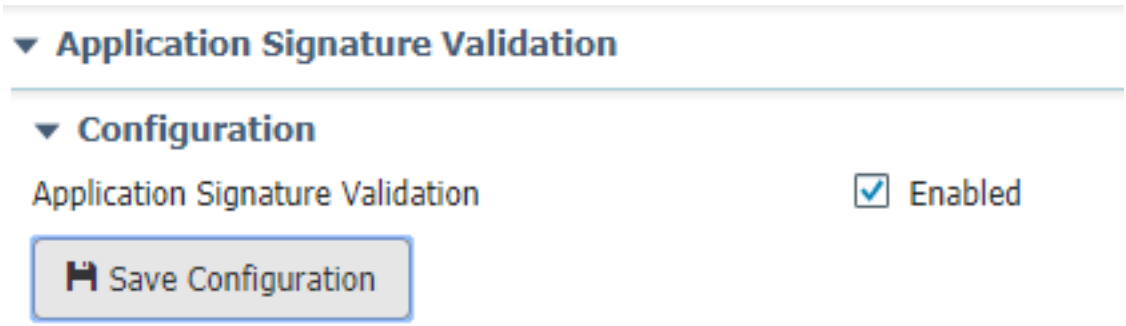
如镜像所显示，导航到**系统设置>导入托拉斯锚点。**



选择如镜像所显示，您在步骤2.生成并且点击OK键的文件。



在您顺利地导入信任锚点后，如镜像所显示，请检查**已启用应用程序签署的验证**并且单击**保存配置**：



步骤4.创建有特殊用途的密钥和CSR

其次，您能创建使用签字到您的IOx应用程序的一个密钥和证书对。最佳实践是生成您计划实施的每应用程序的一特定密钥对。

只要其中每一个签字与同样CA，他们全部考虑作为有效。

为了生成有特殊用途的密钥：

```
[jedepuyd@KJK-SRVIOT-10 signing]$ openssl genrsa -out app-key.pem 2048
```

```
Generating RSA private key, 2048 bit long modulus
.......+++
...+++
e is 65537 (0x10001)
```

为了生成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 []:
```

如同CA，必须调节在应用程序证书的值匹配您的用例。

步骤5.与CA的符号有特殊用途的证书

即然您有您的CA和应用程序CSR的需求，您能签署与使用的CSR CA。结果是一签字的专用证书：

```
[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
```

步骤6.包您的IOx应用程序并且签署它与有特殊用途的证书

这时，您准备包您的IOx应用程序和签署它与从步骤4.的生成的密钥对并且由在步骤5.的CA签了字。

创建来源和package.yaml的进程的其余您的应用程序的依然是不可更改。

包与使用的IOx应用程序密钥对：

```
[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
```

步骤7.部署您的在签名启用的设备上的签字的IOx包

在进程的最后一步是实施应用程序对您的IOx设备。与一个未签名的应用程序部署比较，没有差异：

```
[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
```

验证

使用本部分可确认配置能否正常运行。

为了验证，如果应用程序密钥正确地签字与您的CA，您能执行此：

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

故障排除

本部分提供了可用于对配置进行故障排除的信息。

当您遇到与应用程序的部署的时问题，您可能发现这些错误之一：

```
[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"
}
```

某事在签署出了错与使用的应用程序证书CA或不配比与那个在委托锚点套件。

请使用被提及的说明在Verify部分，检查您的证书并且委托锚点套件。

这些错误表明您的包未正确地签字，您能再调查步骤6。

```
[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"
}
```