在OpenSSL上配置多级CA以生成IOS XE证书

目录

简介

本文档介绍创建多级CA以创建与Cisco IOS® XE设备兼容的通用证书的方法。

先决条件

要求

Cisco 建议您了解以下主题：

- 如何使用OpenSSL应用.
- 公共密钥基础设施(PKI)和数字证书。

使用的组件

本文档中的信息基于以下软件和硬件版本：

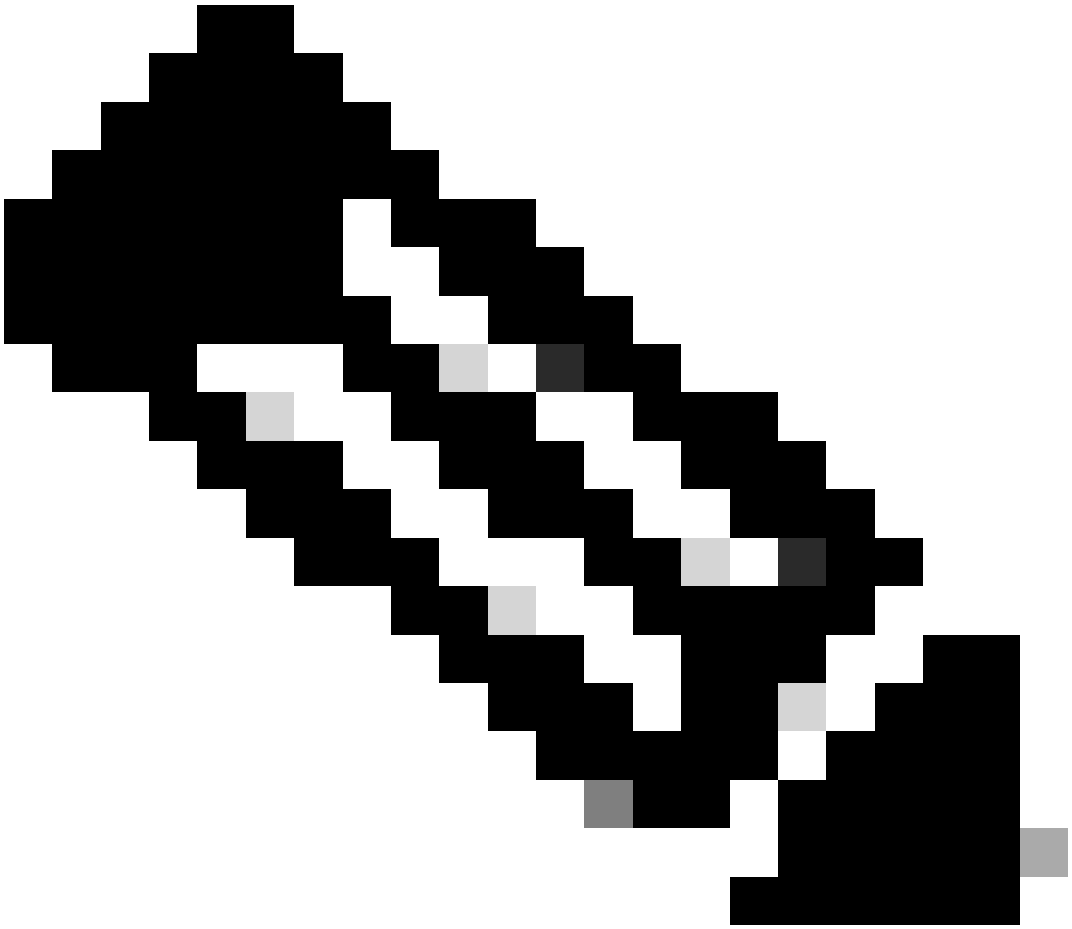
- OpenSSL应用程序（版本3.0.2）。
- 9800 WLC（Cisco IOS XE版本17.12.3）。

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。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。

配置

概述

目的是创建具有根CA和中间CA的两级本地证书颁发机构(CA)来签署设备证书。签名证书后，它们将导入到Cisco IOS XE设备。



> 注意：本文档使用Linux特定命令创建和排列文件。解释这些命令，以便您能够在其它有OpenSSL的操作系统上执行相同的操作。

准备OpenSSL配置文件

在安装了OpenSSL的计算机上从当前工作目录创建名为openssl.conf的文本文件。复制并粘贴这些

行，为OpenSSL提供证书签名所需的配置。您可以根据需要编辑此文件。

```
[ ca ]
default_ca  = IntermCA

[ RootCA ]

dir     = ./RootCA
certs       = $dir/RootCA.db.certs
crl_dir = $dir/RootCA.db.crl
database    = $dir/RootCA.db.index
unique_subject = yes
new_certs_dir   = $dir/RootCA.db.certs
certificate = $dir/RootCA.crt
serial      = $dir/RootCA.db.serial
#crlnumber   = $dir/RootCA.db.crlserial
private_key = $dir/RootCA.key
RANDFILE    = $dir/RootCA.db.rand
name_opt    = ca_default
cert_opt    = ca_default
############################## Modify default days for certificates signed by Root CA (Intermediate certs
default_days    = 360
default_md  = sha256
preserve    = no
policy          = optional_policy

[ IntermCA ]

dir     = ./IntermCA
certs       = $dir/IntermCA.db.certs
crl_dir = $dir/IntermCA.db.crl
database    = $dir/IntermCA.db.index
unique_subject = yes
new_certs_dir   = $dir/IntermCA.db.certs
certificate = $dir/IntermCA.crt
serial      = $dir/IntermCA.db.serial
private_key = $dir/IntermCA.key
RANDFILE    = $dir/IntermCA.db.rand
name_opt    = ca_default
cert_opt    = ca_default
# Certificate field options
############################## Modify default days for certificates signed by Intermediate CA cert (devi
default_days    = 1000
#default_crl_days  = 1000
default_md  = sha256
# use public key default MD
preserve    = no
policy          = optional_policy


[ optional_policy ]
countryName     = optional
stateOrProvinceName = optional
localityName        = optional
organizationName    = optional
organizationalUnitName  = optional
commonName      = supplied

[ req ]
default_bits        = 2048
```

```
default_keyfile     = privkey.pem
distinguished_name  = req_distinguished_name
attributes      = req_attributes
x509_extensions = v3_ca # The extentions to add to the signed cert
string_mask = nombstr

[ req_distinguished_name ]
countryName         = Country Name
countryName_default     = MX
countryName_min         = 2
countryName_max         = 2

stateOrProvinceName     = State or province
stateOrProvinceName_default = CDMX

localityName            = Locality
localityName_default        = CDMX

organizationName      = Organization name
organizationName_default  = Cisco lab

organizationalUnitName      = Organizational unit
organizationalUnitName_default  = Cisco Wireless

commonName          = Common name
commonName_max          = 64



[ req_attributes ]
# challengePassword     = A challenge password
# challengePassword_min     = 4
# challengePassword_max     = 20

#This section contains the extensions used for the Intermediate CA certificate

[ v3_ca ]
# Extensions for a typical CA
basicConstraints = CA:true
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer:always
subjectAltName = @Intermediate_alt_names



[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth, clientAuth

[ crl_ext ]
# CRL extensions.
#authorityKeyIdentifier=keyid:always,issuer:always


#DEFINE HERE SANS/IPs NEEDED for Intermediate CA device certificates
[Intermediate_alt_names]
DNS.1   = Intermediate.example.com
DNS.2   = Intermediate2.example.com


#Section for endpoint certificate CSR generation
```

```
[ endpoint_req_ext ]
subjectAltName = _alt_names

#Section for endpoint certificate sign by CA
[ Endpoint ]
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
#Change the key usage according to the certificate usage needs
extendedKeyUsage = clientAuth
subjectAltName = _alt_names

#Define here SANS/IPs needed for  Endpoint certificates
[endpoint_alt_names]
DNS.1   = Endpoint.example.com
DNS.2   = Endpoint2.example.com

#Section for IOS-XE device certificate CSR generation
[ device_req_ext ]
subjectAltName = @IOS_alt_names

#Section for IOS-XE certificate sign by CA
[ IOS_cert ]
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
#Change the key usage according to the certificate usage needs
extendedKeyUsage = clientAuth , serverAuth
subjectAltName = @IOS_alt_names

#Define here SANS/IPs needed for IOS-XE certificates
[IOS_alt_names]
DNS.1   = IOSXE.example.com
DNS.2   = IOSXE2.example.com
```

为证书颁发机构创建初始文件

在当前目录下创建一个名为RootCA的文件夹。在其中，再创建名为RootCA.db.tmp、RootCA.db.certs和RootCA.db.crl的3个文件夹。

```
mkdir RootCA
mkdir RootCA/RootCA.db.tmp
mkdir RootCA/RootCA.db.certs
mkdir RootCA/RootCA.db.crl
```

在RootCA文件夹中创建名为RootCA.db.serial的文件。此文件需要包含证书序列号的初始值，01是此情况下选择的值。

在RootCA文件夹中创建名为RootCA.db.crlserial的文件。此文件需要包含证书撤销列表编号的初始值，在此案例中选定的值为01。

```
echo 01 > RootCA/RootCA.db.serial
echo 01 > RootCA/RootCA.db.crlserial
```

在RootCA文件夹中创建名为RootCA.db.index的文件。

```
touch RootCA/RootCA.db.index
```

在RootCA文件夹中创建名为RootCA.db.rand的文件，并用8192个随机字节填充该文件，以用作内部随机数生成器的种子。

```
openssl rand -out RootCA/RootCA.db.rand 8192
```

在当前目录下创建一个名为IntermCA的文件夹。在其中，再创建名为IntermCA.db.tmp、IntermCA.db.certs和IntermCA.db.crl的3个文件夹。

```
mkdir IntermCA
mkdir IntermCA/IntermCA.db.tmp
mkdir IntermCA/IntermCA.db.certs
mkdir IntermCA/IntermCA.db.crl
```

在IntermCA文件夹中创建名为IntermCA.db.serial的文件。此文件需要包含证书序列号的初始值，01是此情况下选择的值。

在IntermCA文件夹中创建名为IntermCA.db.crlserial的文件。此文件需要包含证书撤销列表编号的初始值，在此案例中选定的值为01。

```
echo 01 > IntermCA/IntermCA.db.serial
echo 01 > IntermCA/IntermCA.db.crlserial
```

在IntermCA文件夹中创建名为IntermCA.db.index的文件。

在IntermCA文件夹中创建名为IntermCA.db.rand的文件，并用8192个随机字节填充该文件，以用作内部随机数生成器的种子。

```
touch IntermCA/IntermCA.db.index
```

在IntermCA文件夹中创建名为IntermCA.db.rand的文件，并用8192个随机字节填充该文件，以用作内部随机数生成器的种子。

```
openssl rand -out IntermCA/IntermCA.db.rand 8192
```

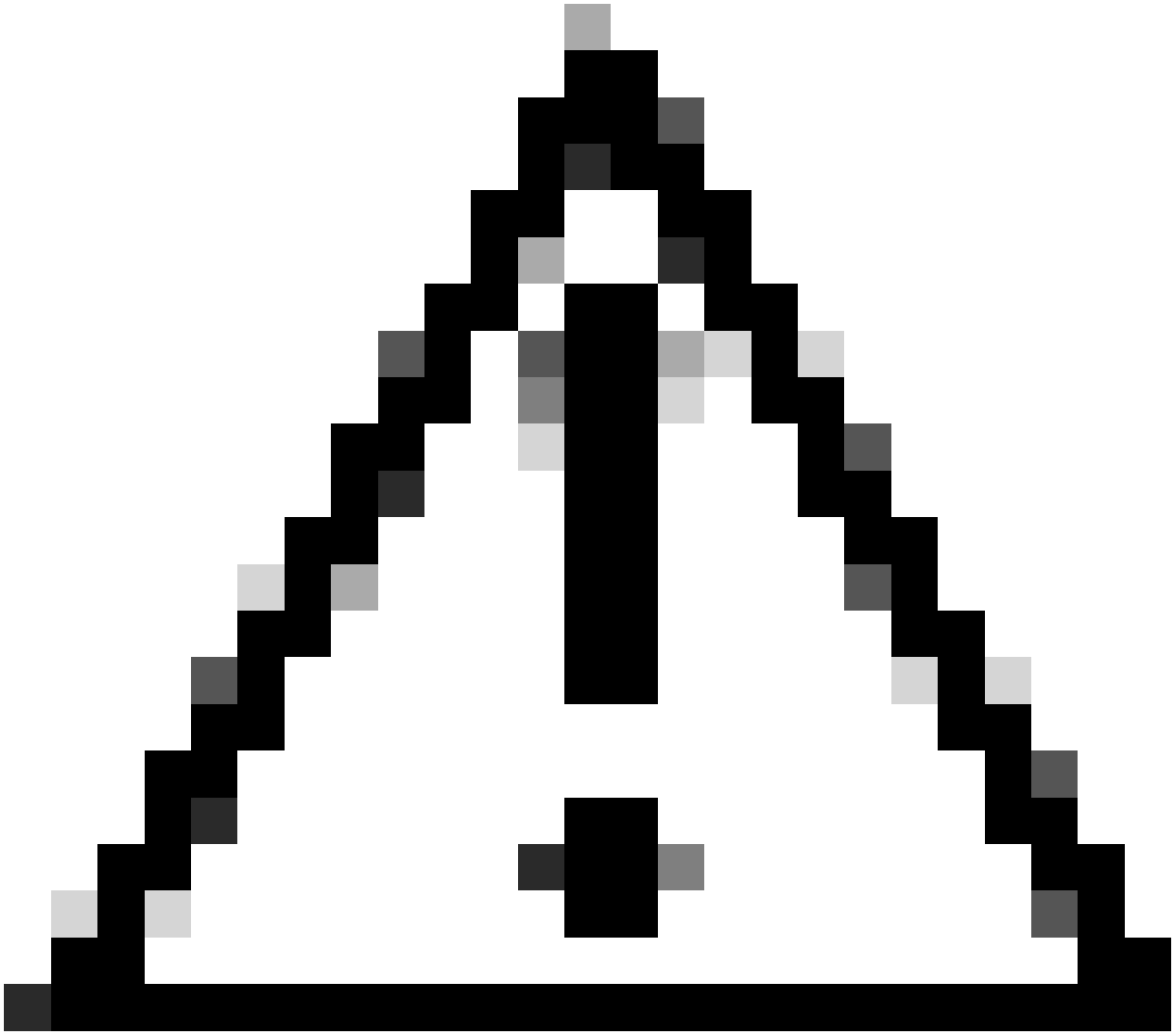
这是创建所有初始Root和中间CA文件后的文件结构。

```
mariomed@CSCO-W-PF320YP6:/mnt/c/Users/mariomed/radsecfiles1$ tree
.
├── IntermCA
│   ├── IntermCA.db.certs
│   ├── IntermCA.db.crl
│   ├── IntermCA.db.crlserial
│   ├── IntermCA.db.index
│   ├── IntermCA.db.rand
│   ├── IntermCA.db.serial
│   └── IntermCA.db.tmp
├── RootCA
│   ├── RootCA.db.certs
│   ├── RootCA.db.crl
│   ├── RootCA.db.crlserial
│   ├── RootCA.db.index
│   ├── RootCA.db.rand
│   ├── RootCA.db.serial
│   └── RootCA.db.tmp
└── openssl.cnf
```

创建根CA证书

运行此命令可为根CA创建私钥。

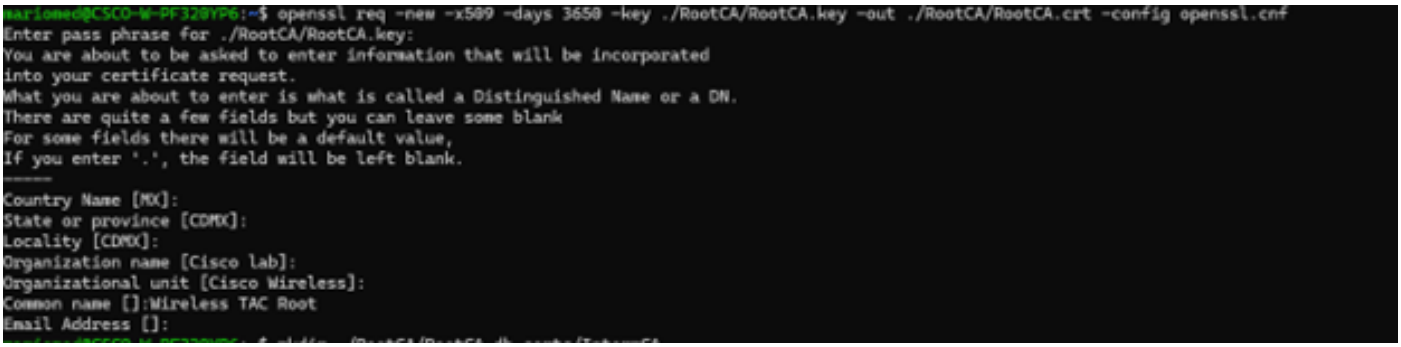
```
openssl genrsa -des3 -out ./RootCA/RootCA.key 4096
```



注意：生成密钥时，OpenSSL要求您提供口令。将密码短语机密和生成的私钥保存在安全位置。任何有权访问该证书的人都可以颁发证书作为您的根CA。

在openSSL上使用req命令创建根CA自签名证书。-x509标志在内部创建证书签名请求(CSR)并自动对其进行自签名。编辑-days参数和主题备用名称。perminal会提示您提供一个公用名称。确保您输入的公用名称与主题备用名称(SAN)匹配。

```
openssl req -new -key ./RootCA/RootCA.key -out ./RootCA/RootCA.crt -config openssl.cnf -x509 -days 3650
```



OpenSSL可分辨名称交互式提示

生成的文件名为RootCA.crt，位于RootCA文件夹中。此文件是根CA证书。

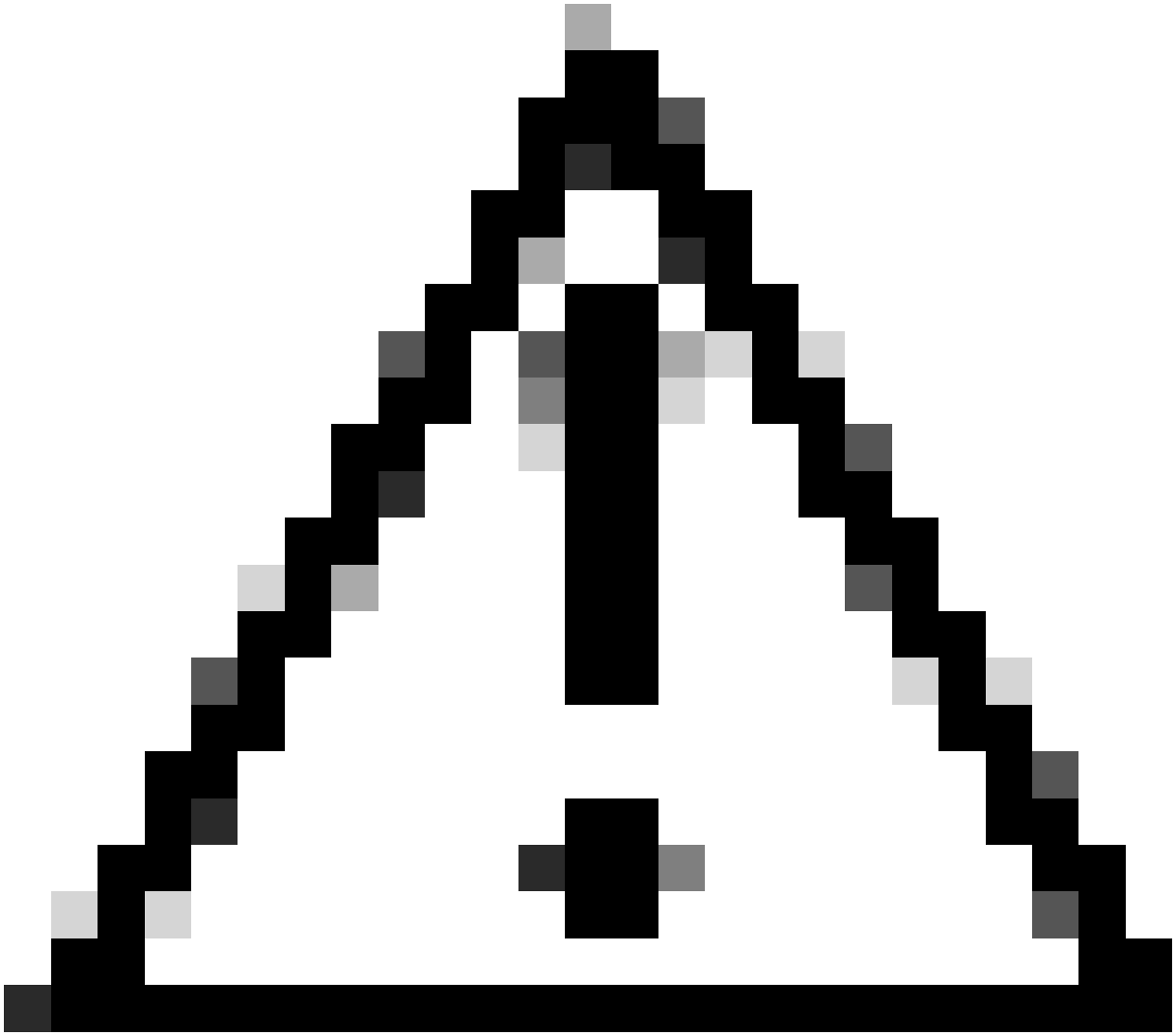
创建中间CA证书

创建文件夹以将签名中间CA证书存储在根文件夹中。

```
mkdir ./RootCA/RootCA.db.certs/IntermCA
```

为中间证书创建私钥。

```
openssl genrsa -des3 -out ./RootCA/RootCA.db.certs/IntermCA/IntermCA.key 4096
```



注意：生成密钥时，OpenSSL要求您提供口令。将密码短语机密和生成的私钥保存在安全位置。任何有权访问该证书的人都可以颁发证书作为您的中间CA。

创建中间CA证书签名请求。终端会提示您输入证书信息。

```
openssl req -new -key ./RootCA/RootCA.db.certs/IntermCA/IntermCA.key -out ./RootCA/RootCA.db.certs/Inte
```

使用openssl.cnf文件的RootCA部分为中间CSR签名。

```
openssl ca -config openssl.cnf -name RootCA -extensions v3_ca -out ./RootCA/RootCA.db.certs/IntermCA/In
```

生成的文件名为IntermCA.crt，位于RootCA文件夹内。此文件是根CA证书。

将中间证书和密钥移动到您作为中间CA的初始文件的一部分创建的其自己的文件夹。

```
cp ./RootCA/RootCA.db.certs/IntermCA/IntermCA.crt ./RootCA/RootCA.db.certs/IntermCA/IntermCA.key ./Inte
```

这是为初始根和中间CA创建私钥和证书之后的文件结构。

```
mariomed@CSCO-W-PF320YP6:/mnt/c/Users/mariomed/radsecfiles$ tree
.
├── IntermCA
│   ├── IntermCA.crt  <------Intermediate CA certficate
│   ├── IntermCA.db.certs
│   ├── IntermCA.db.crl
│   ├── IntermCA.db.crlserial
│   ├── IntermCA.db.index
│   ├── IntermCA.db.rand
│   ├── IntermCA.db.serial
│   ├── IntermCA.db.tmp
│   └── IntermCA.key  <------Intermediate CA private key
├── RootCA
│   ├── RootCA.crt    <------Root CA certficate
│   ├── RootCA.db.certs
│   │   ├── 01.pem
│   │   └── IntermCA
│   │       ├── IntermCA.crt
│   │       ├── IntermCA.csr
│   │       └── IntermCA.key
│   ├── RootCA.db.crl
│   ├── RootCA.db.crlserial
│   ├── RootCA.db.index
│   ├── RootCA.db.index.attr
│   ├── RootCA.db.index.old
│   ├── RootCA.db.rand
│   ├── RootCA.db.serial
│   ├── RootCA.db.serial.old
│   ├── RootCA.db.tmp
│   └── RootCA.key    <------Root CA private key
└── openssl.cnf
```

创建设备证书

创建Cisco IOS XE设备证书

创建新文件夹以存储Cisco IOS XE设备证书。

```
mkdir ./IntermCA/IntermCA.db.certs/IOSdevice
```

创建设备私钥IOSdevice.key和设备CSR IOSdevice.csr。使用device_req_ext部分将上述部分下的SAN添加到CSR中。

```
openssl req -newkey rsa:4096 -sha256 -keyout ./IntermCA/IntermCA.db.certs/IOSdevice/IOSdevice.key -node
```

修改opnessl.cnf文件[IOS_alt_names]部分，以便在CSR上提供的公用名与SAN匹配。

```
#Define here SANS/IPs needed for IOS-XE certificates
[IOS_alt_names]
DNS.1   = IOSXE.example.com
DNS.2   = IOSXE2.example.com
```

使用中间CA IntermCA部分签署IOS XE设备CSR。使用-config指向openSSL配置文件，并使用-extensions指向IOS_cert部分。这样，SAN将保留在签名证书上。

```
openssl ca -config openssl.cnf -extensions IOS_cert -name IntermCA -out ./IntermCA/IntermCA.db.certs/IO
```

完成此步骤后，您为名为IOSdevice.crt的IOS XE设备创建了具有匹配私钥IOSdevice.key的有效证书。

可选-创建终端证书

此时，您已部署本地CA并为IOS XE设备颁发了一个证书。您还可以使用此CA生成终端身份证书。这些证书也有效，例如，在9800无线局域网控制器上执行本地EAP身份验证，甚至使用RADIUS服务器执行dot1x身份验证。此部分帮助您生成终端证书。

创建用于存储终端证书的文件夹。

```
mkdir ./IntermCA/IntermCA.db.certs/Endpoint
```

修改openSSL.cnf文件[endpoint_alt_names]部分，以便在CSR上提供的公用名与SAN匹配。

```
#Define here SANS/IPs needed for  Endpoint certificates
[endpoint_alt_names]
```

```
DNS.1   = Endpoint.example.com
DNS.2   = Endpoint2.example.com
```

使用用于SAN的endpoint_req_ext部分创建终端私钥和WLC CSR。

```
openssl req -newkey rsa:2048 -keyout ./IntermCA/IntermCA.db.certs/Endpoint/Endpoint.key -nodes -config
```

签署终端设备证书。

```
openssl ca -config openssl.cnf -extensions Endpoint -name IntermCA -out ./IntermCA/IntermCA.db.certs/En
```

将证书导入到Cisco IOS XE设备

根据导入到Cisco IOS XE设备所需的步骤，在同一文件中创建包含根CA和中间CA的文件，并将其保存至名为certfile.crt的./IntermCA/IntermCA.db.certs/WLC/文件夹。

```
cat ./RootCA/RootCA.crt ./IntermCA/IntermCA.crt > ./IntermCA/IntermCA.db.certs/IOSdevice/certfile.crt
```

9800系列WLC使用不同的命令创建用于证书导入的pfx文件。要创建pfx文件，请根据Cisco IOS XE版本运行以下命令之一。

有关证书导入过程的详细信息，请参阅在Catalyst 9800 WLC上生成和下载CSR证书

对于早于17.12.1的版本：

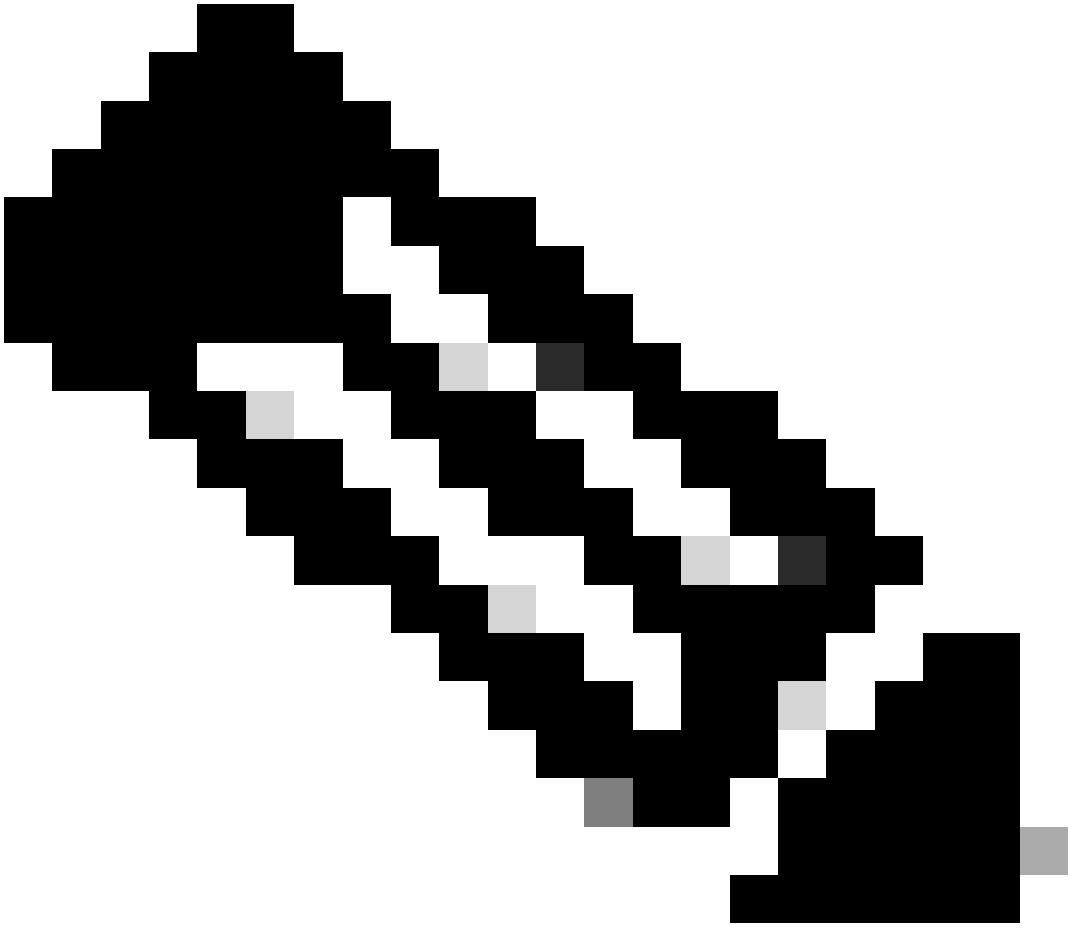
```
openssl pkcs12 -export -macalg sha1 -legacy -descert -out ./IntermCA/IntermCA.db.certs/IOSdevice/IOSdev
```

对于版本17.12.1或更高版本：

```
openssl pkcs12 -export -out ./IntermCA/IntermCA.db.certs/IOSdevice/IOSdevice.pfx -inkey ./IntermCA/Inte
```

将IOSdevice.pfx证书导入到Cisco IOS XE设备：

```
WLC# configure terminal
WLC(config)#crypto pki import




      pkcs12 [tftp://




       /




        | ftp://




         /




          | http://




           /




            | bootflash:




             ] password
```



注意：确保需要验证设备证书的设备信任为本指南创建的CA证书。例如，如果设备证书用于Cisco IOS XE设备上的Web管理目的，则访问管理员门户的任何计算机或浏览器都需要在其信任库上具有CA证书。

禁用证书的吊销检查，因为Cisco IOS XE设备可以从已部署的CA检查没有联机证书吊销列表。您必须在作为验证路径一部分的所有信任点上禁用它。根CA信任点与中间/设备信任点具有相同的名称，并在末尾附加了字符串-rrr1。

```
9800#configure terminal

9800(config)#crypto pki trustpoint IOSdevice.pfx
9800(config)#revocation-check none
9800(config)#exit

9800(config)#crypto pki trustpoint IOSdevice.pfx-rrr1
9800(config)#revocation-check none
9800(config)#exit
```

验证

验证OpenSSL上的证书信息

要验证已创建证书的证书信息，请在Linux终端上运行命令：

```
openssl x509 -in

      -text -noout
```

显示完整的证书信息。

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 2 (0x2)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = MX, ST = CDMX, L = CDMX, O = Cisco lab, OU = Cisco Wireless, CN = Intermediate.example.com
        Validity
            Not Before: Jul 18 19:14:57 2024 GMT
            Not After : Apr 14 19:14:57 2027 GMT
        Subject: C = MX, ST = CDMX, L = CDMX, O = Cisco lab, OU = Cisco Wireless, CN = WLC.example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:b1:10:7d:6c:6c:14:2f:18:a6:0b:69:d9:60:03:
                    56:2d:48:22:f0:42:10:65:44:24:3b:54:e1:4b:87:
                    b8:ab:c5:5f:f6:a1:a3:5e:f6:3c:c5:45:cc:01:6d:
                    df:e8:a7:81:28:50:44:54:4c:af:a0:56:cf:06:be:
                    10:7e:e2:46:42:ea:3c:b9:d4:03:75:08:84:70:36:
                    bb:3d:95:3b:e2:86:e6:f7:d9:4d:00:28:c4:3c:cb:
                    f8:6d:37:5c:89:28:c1:75:b1:7e:fa:bd:91:cf:8e:
                    5c:a2:37:4f:71:da:6a:04:ee:ba:68:bf:4d:f2:d3:
                    ae:aa:13:42:3b:ff:a0:b3:65:c9:ff:f6:9a:06:d7:
                    6c:08:10:e0:b9:d8:ca:93:2d:e5:5d:7b:74:cd:93:
                    68:b1:46:c7:35:d7:6b:0f:a6:ae:34:e6:23:d1:c8:
                    d3:bf:c0:85:ab:2d:02:a8:dd:54:77:e3:32:61:4e:
                    33:58:b0:62:12:82:42:ae:2b:69:f0:5f:0c:90:c7:
                    9c:ef:b9:9c:fc:29:e2:2c:cb:b4:a9:01:fa:5d:3c:
                    97:11:67:cc:25:96:01:3d:26:1a:43:34:bd:43:b0:
                    a0:f1:ec:a0:c7:98:ad:32:32:99:9c:6b:61:af:57:
                    53:ee:20:cc:d5:ed:db:1c:5c:65:51:42:8c:28:bf:
                    62:bf
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Key Identifier:
                87:89:CA:28:06:95:D5:CE:7C:66:B4:75:81:AA:D4:19:EC:43:01:BB
            X509v3 Authority Key Identifier:
                keyid:2B:08:D8:4C:23:72:5B:62:03:EA:44:F6:9E:D9:F7:75:2E:64:97:DE
                DirName:/C=MX/ST=CDMX/L=CDMX/O=Cisco lab/OU=Cisco Wireless/CN=RootCA
                serial:01
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Subject Alternative Name:
                DNS:WLC.example.com, DNS:WLC2.example.com
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
```

OpenSSL所示的Cisco IOS XE设备证书信息

验证Cisco IOS XE设备上的证书信息。

命令show crypto pki certificates verbose可以打印设备上所有可用证书的证书信息。

```
9800#show crypto pki certificates verbose
CA Certificate <------Type of certificate
  Status: Available
  Version: 3
  Certificate Serial Number (hex): 2A352E27C69021ECE1AA61751CA1F233E0636FB1
  Certificate Usage: General Purpose
  Issuer: <-------------DN for issuer
    cn=RootCA
    ou=Cisco Wireless
    o=Cisco lab
    l=CDMX
    st=CDMX
    c=MX
```

```
  Subject: <--------------DN for subject
    cn=RootCA
    ou=Cisco Wireless
    o=Cisco lab
    l=CDMX
    st=CDMX
    c=MX
  Validity Date:  <--------Validity date
    start date: 14:54:02 Central Jul 22 2024
    end   date: 14:54:02 Central Jul 20 2034
  Subject Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (2048 bit)  <--------------Key size
  Signature Algorithm: SHA256 with RSA Encryption
  Fingerprint MD5: 432021B5 B4BE15F5 A537385C 4FAB9A94
  Fingerprint SHA1: 86D18427 BE619A2A 6C20C314 9EDAAEB2 6B4DFE87
  X509v3 extensions:
    X509v3 Subject Key ID: 57DEEBD8 3214CA05 176F0CD6 6C842EBC 9ABFF7D8
    X509v3 Basic Constraints:
        CA: TRUE
    X509v3 Subject Alternative Name:
        RootCA  <------------SANs
        IP Address :
        OtherNames :
    X509v3 Authority Key ID: 57DEEBD8 3214CA05 176F0CD6 6C842EBC 9ABFF7D8
    Authority Info Access:
  Cert install time: 16:42:09 Central Jul 22 2024
  Associated Trustpoints: WLC.pfx-rrr1  <----------Associated trustpoint
  Storage: nvram:RootCA#6FB1CA.cer
```

故障排除

已进行吊销检查

当证书导入到Cisco IOS XE时，新创建的信任点启用撤销检查。如果向需要使用导入的证书信任点进行验证的设备提供证书，则设备会搜索不存在的证书撤销列表，并且会失败。消息会打印在终端上。

```
Jul 17 21:50:39.068: %PKI-3-CRL_FETCH_FAIL: CRL fetch for trustpoint WLC1.pfx failed
                      Reason : Enrollment URL not configured.
```

确保证书的验证路径中的每个信任点都包含命令revocation-check none。

相关信息

- 在Catalyst 9800 WLC上生成和下载CSR证书
- 使用IOS XE PKI配置CA签名证书
- 安全与VPN配置指南，Cisco IOS XE 17.x

- 了解证书信息，为9800 WLC创建链

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。