OpenSSL에서 다중 레벨 CA를 구성하여 IOS XE 인증서 생성

목차

소개

이 문서에서는 Cisco IOS® XE 디바이스와 호환되는 범용 인증서를 생성하기 위해 다중 레벨 CA를 생성하는 방법에 대해 설명합니다.

사전 요구 사항

요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- OpenSSL 애플리케이션 사용 방법
- PKI(Public Key Infrastructure) 및 디지털 인증서

사용되는 구성 요소

이 문서의 정보는 다음 소프트웨어 및 하드웨어 버전을 기반으로 합니다.

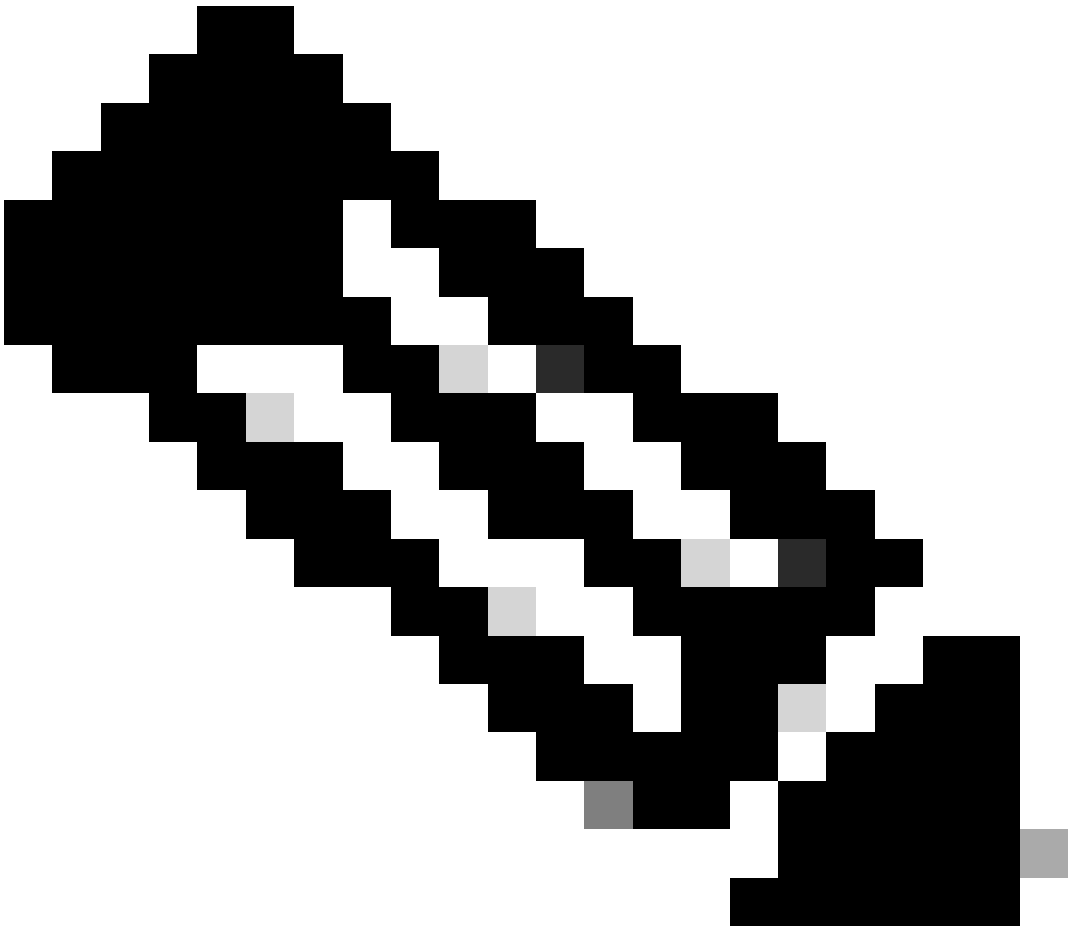
- OpenSSL 애플리케이션(버전 3.0.2).
- 9800 WLC(Cisco IOS XE 버전 17.12.3).

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

구성

개요

루트 CA와 중간 CA가 있는 두 가지 레벨의 로컬 CA(Certificate Authority)를 생성하여 디바이스 인증서에 서명하는 것입니다. 인증서가 서명되면 Cisco IOS XE 디바이스로 가져옵니다.



참고: 이 문서에서는 Linux 관련 명령을 사용하여 파일을 만들고 정렬합니다. OpenSSL을 사용할 수 있는 다른 운영 체제에서도 동일한 작업을 수행할 수 있도록 이 명령에 대해 설명합니다.

OpenSSL 컨피그레이션 파일 준비

OpenSSL이 설치된 시스템의 현재 작업 디렉토리에서 openssl.conf라는 텍스트 파일을 생성합니다. OpenSSL에 인증서 서명에 필요한 컨피그레이션을 제공하려면 이러한 행을 복사하여 붙여넣습니다. 필요에 맞게 이 파일을 수정할 수 있습니다.

```
[ 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 cert
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
```

인증 기관에 대한 초기 파일 만들기

현재 디렉터리에 RootCA라는 폴더를 만듭니다. 그 안에 RootCA.db.tmp, RootCA.db.certs, RootCA.db.crl이라는 3개의 폴더를 만듭니다.

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

RootCA 폴더 내에 RootCA.db.serial이라는 파일을 만듭니다. 이 파일은 인증서 일련 번호의 초기 값을 포함해야 합니다. 01은 이 경우에 선택한 값입니다.

RootCA 폴더 내에 RootCA.db.crlserial이라는 파일을 생성합니다. 이 파일은 인증서 폐기 목록 번호의 초기 값을 포함해야 합니다. 01은 이 경우에 선택한 값입니다.

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

RootCA 폴더 내에 RootCA.db.index라는 파일을 생성합니다.

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

RootCA 폴더 내에 RootCA.db.rand라는 파일을 만들고 8192 임의 바이트로 채워 내부 난수 생성기의 시드로 사용합니다.

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

현재 디렉토리에 IntermCA라는 폴더를 생성합니다. 그 안에 IntermCA.db.tmp, IntermCA.db.certs, IntermCA.db.crl이라는 3개의 폴더를 만듭니다.

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

IntermCA 폴더 내에 IntermCA.db.serial이라는 파일을 만듭니다. 이 파일은 인증서 일련 번호의 초기 값을 포함해야 합니다. 01은 이 경우에 선택한 값입니다.

IntermCA 폴더 내에 IntermCA.db.crlserial이라는 파일을 생성합니다. 이 파일은 인증서 폐기 목록 번호의 초기 값을 포함해야 합니다. 01은 이 경우에 선택한 값입니다.

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

IntermCA 폴더 내에 IntermCA.db.index라는 파일을 생성합니다.

IntermCA 폴더 내에 IntermCA.db.rand라는 파일을 만들고 8192 랜덤 바이트로 채워 내부 난수 생성기의 시드 역할을 합니다.

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

IntermCA 폴더 내에 IntermCA.db.rand라는 파일을 만들고 8192 랜덤 바이트로 채워 내부 난수 생성기의 시드 역할을 합니다.

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

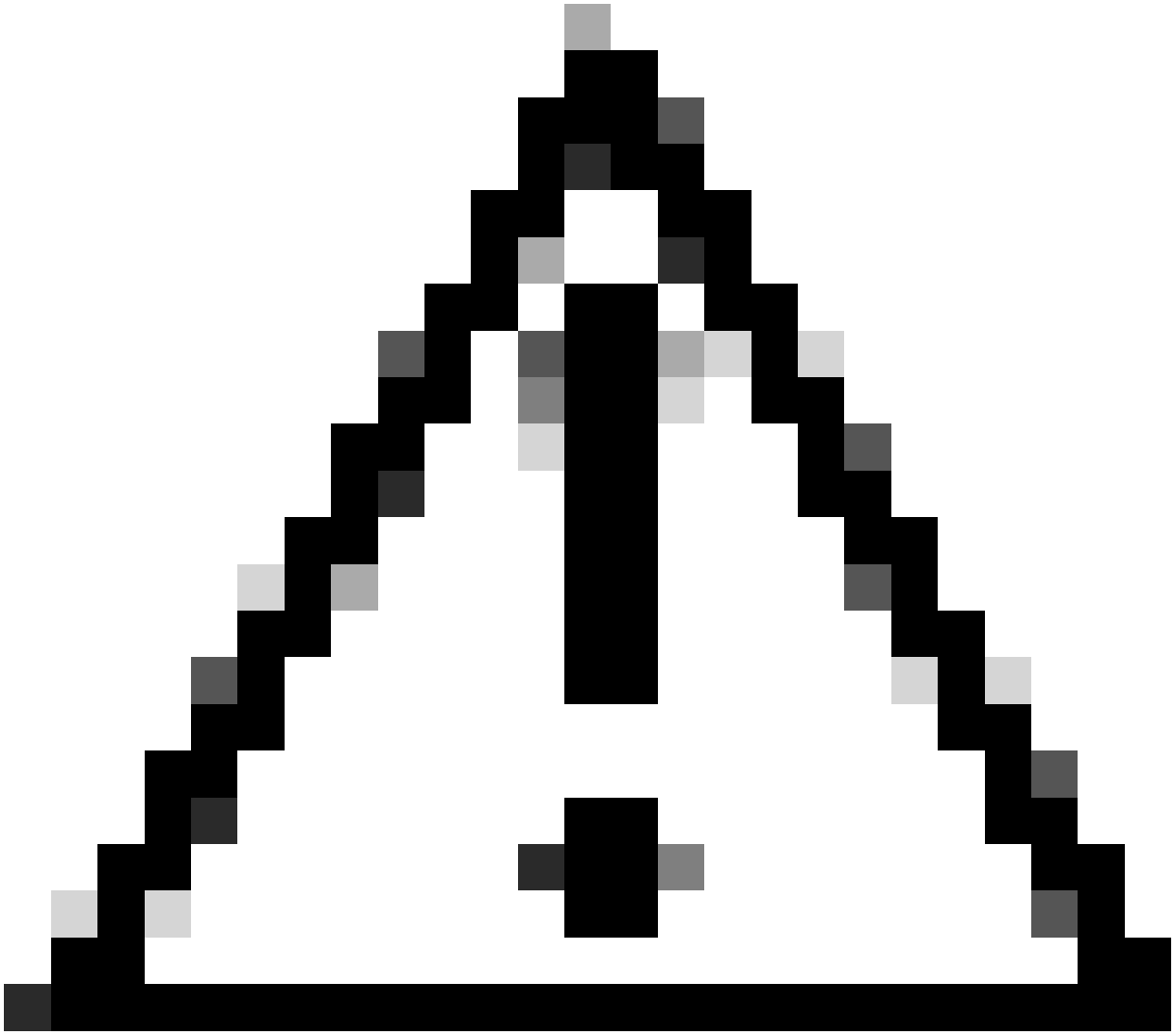
이것은 모든 초기 루트 및 중간 CA 파일을 생성한 후의 파일 구조입니다.

```
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
```

루트 CA 인증서 생성

이 명령을 실행하여 루트 CA의 개인 키를 생성합니다.

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

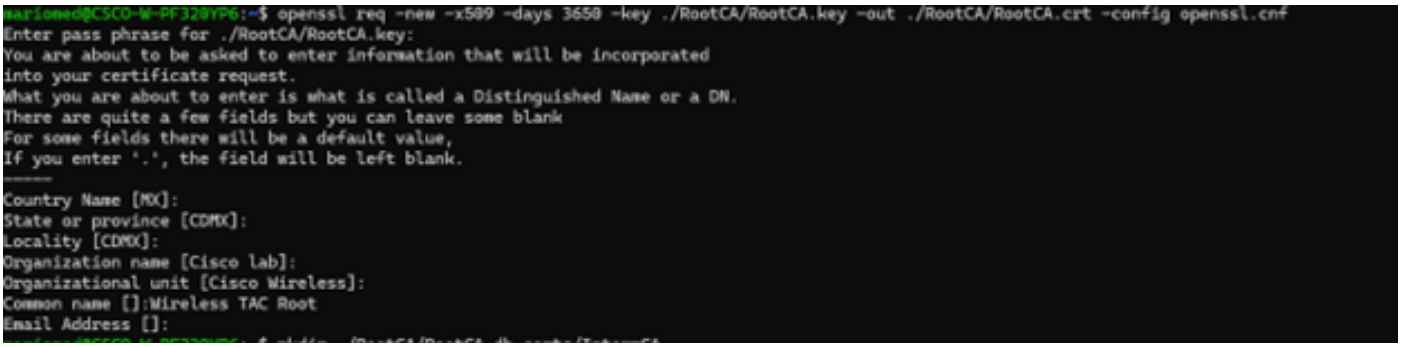


주의: OpenSSL에서는 키를 생성할 때 암호를 제공해야 합니다. 암호 비밀과 생성된 개인 키를 안전한 위치에 보관합니다. 액세스 권한이 있는 사용자는 인증서를 루트 CA로 발급할 수 있습니다.

---

openSSL에서 명령을 사용하여 루트 CA 자체 서명 인증서를req생성합니다. 플래그는 -x509 내부적으로 CSR(Certificate Signing Request)을 생성하고 자동으로 자체 서명합니다. 매개변수-days및 주체 대체 이름을 편집합니다. 터미널은 일반 이름을 입력하라는 메시지를 표시합니다. 입력한 일반 이름이 SAN(주체 대체 이름)과 일치하는지 확인합니다.

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



OpenSSL 고유 이름 인터랙티브 프롬프트

생성된 파일은 RootCA.crt라고 하며 RootCA 폴더 내에 있습니다. 이 파일은 루트 CA 인증서입니다 .

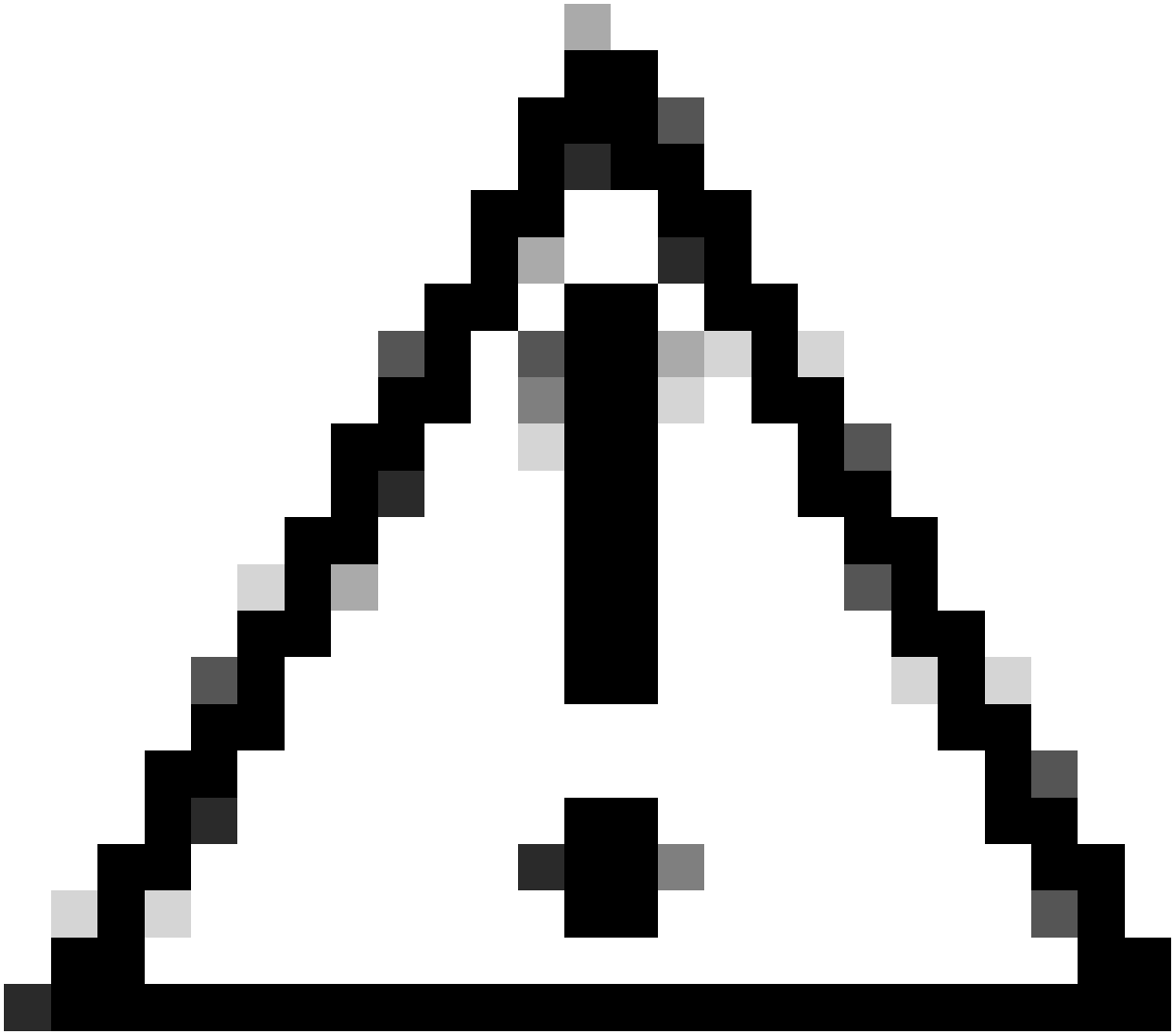
중간 CA 인증서 생성

서명된 중간 CA 인증서를 루트 폴더 내에 저장할 폴더를 만듭니다.

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

중간 인증서에 대한 개인 키를 만듭니다.

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



주의: OpenSSL에서는 키를 생성할 때 암호를 제공해야 합니다. 암호 비밀과 생성된 개인 키를 안전한 위치에 보관합니다. 액세스 권한이 있는 사용자는 누구나 중간 CA로 인증서를 발급할 수 있습니다.

중간 CA 인증서 서명 요청을 생성합니다. 터미널은 인증서 정보를 입력하라는 메시지를 표시합니다.

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

openssl.cnf 파일의 RootCA 섹션을 사용하여 중간 CSR에 서명합니다.

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

생성된 파일은 IntermCA.crt라고 하며 RootCA 폴더 내에 있습니다. 이 파일은 루트 CA 인증서입니다.

중간 인증서와 키를 중간 CA의 초기 파일의 일부로 생성한 고유 폴더로 이동합니다.

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

이는 초기 루트 및 중간 CA에 대한 프라이빗 키 및 인증서를 생성한 후의 파일 구조입니다.

```
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
```

디바이스 인증서 생성

Cisco IOS XE 디바이스 인증서 생성

Cisco IOS XE 디바이스 인증서를 저장할 새 폴더를 생성합니다.

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

디바이스 개인 키 IOSdevice.key 및 디바이스 CSR IOSdevice.csr을 생성합니다. device_req_ext 섹션을 사용하여 해당 섹션의 SAN을 CSR에 추가합니다.

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

CSR에서 제공하는 공통 이름이 SAN과 일치하도록 opnessl.cnf 파일 [IOS_alt_names] 섹션을 수정합니다.

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

중간 CA IntermCA 섹션으로 IOS XE 디바이스 CSR에 서명합니다. openSSL 컨피그레이션 파일-config을 가리키고 IOS_cert 섹션-extensions을 가리키려면 를 사용합니다. 이렇게 하면 서명된 인증서에 SAN이 유지됩니다.

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

이 단계 후 일치하는 개인 키 IOSdevice.key를 사용하여 IOSdevice.crt라는 IOS XE 디바이스에 대한 유효한 인증서를 생성했습니다.

선택 사항 - 엔드포인트 인증서 생성

이 시점에서 로컬 CA를 구축하고 IOS XE 디바이스용 인증서 하나를 발급했습니다. 이 CA를 사용하여 엔드포인트 ID 인증서를 생성할 수도 있습니다. 이러한 인증서는 예를 들어 9800 Wireless LAN Controller에서 로컬 EAP 인증을 수행하거나 RADIUS 서버를 통한 dot1x 인증도 수행할 수 있습니다. 이 섹션에서는 엔드포인트 인증서를 생성하는 데 도움이 됩니다.

엔드포인트 인증서를 저장할 폴더를 생성합니다.

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

CSR에서 제공하는 공통 이름이 SAN과 일치하도록 openSSL.cnf 파일 [endpoint_alt_names]섹션을 수정합니다.

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

SAN용 endpoint_req_ext 섹션을 사용하여 엔드포인트 개인 키 및 WLC CSR을 생성합니다.

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

엔드 포인트 장치 인증서를 서명 합니다.

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

Cisco IOS XE 디바이스에 인증서 가져오기

동일한 파일에 루트 CA 및 중간 CA가 포함된 파일을 생성하고 Cisco IOS XE 디바이스로 가져오는 데 필요한 대로 certfile.crt라는 이름을 사용하여 ./IntermCA/IntermCA.db.certs/WLC/폴더에 저장합니다.

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

9800 Series WLC는 인증서 가져오기를 위해 pfx 파일을 생성하기 위해 다양한 명령을 사용합니다. pfx 파일을 만들려면 Cisco IOS XE 버전에 따라 다음 명령 중 하나를 실행합니다.

인증서 가져오기 프로세스에 대한 자세한 내용은 Catalyst 9800 WLC에서 CSR 인증서 생성 및 다운로드를 참조하십시오

17.12.1 이전 버전의 경우:

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

버전 17.12.1 이상의 경우:

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

IOSdevice.pfx 인증서를 Cisco IOS XE 디바이스로 가져옵니다.

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

```
      pkcs12 [tftp://
```

```
       /
```

```
        | ftp://
```

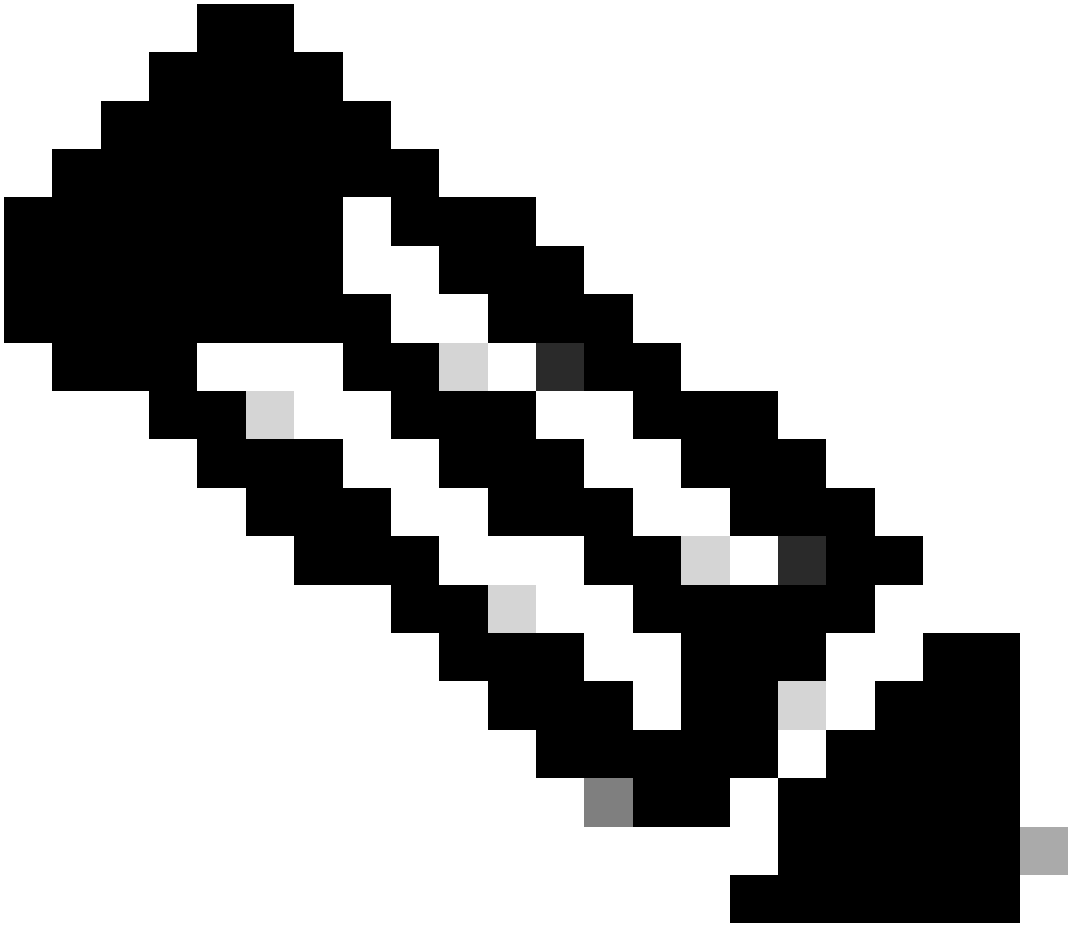
```
         /
```

```
          | http://
```

```
           /
```

```
            | bootflash:
```

] password



참고: 이 설명서에 대해 생성된 CA 인증서가 디바이스 인증서를 검증해야 하는 디바이스에서 신뢰되는지 확인하십시오. 예를 들어, 디바이스 인증서가 Cisco IOS XE 디바이스에서 웹 관리를 위해 사용되는 경우 관리 포털에 액세스하는 모든 컴퓨터 또는 브라우저는 트러스트 스토어에 CA 인증서가 있어야 합니다.

구축한 CA에서 Cisco IOS XE 디바이스가 확인할 수 있는 온라인 인증서 폐기 목록이 없으므로 인증서에 대한 폐기 검사를 비활성화합니다.
확인 경로의 일부인 모든 신뢰 지점에서 비활성화해야 합니다. 루트 CA 신뢰 지점은 중간/디바이스 신뢰 지점과 이름이 동일하며 끝에 문자열 -rrr1이 추가됩니다.

```
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
```

다음을 확인합니다.

OpenSSL에서 인증서 정보 확인

생성된 인증서에 대한 인증서 정보를 확인하려면 Linux 터미널에서 다음 명령을 실행합니다.

```
openssl x509 -in

      -text -noout
```

전체 인증서 정보를 표시합니다.

```
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:
```

Cisco IOS XE 디바이스 인증서 정보(OpenSSL 표시)

Cisco IOS XE 디바이스에서 인증서 정보를 확인합니다.

명령은show crypto pki certificates verbose디바이스에서 사용 가능한 모든 인증서의 인증서 정보를 인쇄합니다.

```
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
```

문제 해결

폐기 검사 진행 중

인증서를 Cisco IOS XE로 가져올 때 새로 생성된 신뢰 지점에는 폐기 검사가 활성화되어 있습니다. 검증을 위해 가져온 인증서 신뢰 지점을 사용해야 하는 디바이스에 인증서가 제시되면 디바이스는 존재하지 않는 인증서 해지 목록을 검색하고 실패합니다. 터미널에 메시지가 출력됩니다.

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

인증서에 대한 확인 경로의 각 신뢰 지점에 명령이 포함되어 있는지 확인합니다`revocation-check none`.

관련 정보

- Catalyst 9800 WLC에서 CSR 인증서 생성 및 다운로드
- IOS XE PKI를 사용하여 CA 서명 인증서 구성

- 보안 및 VPN 컨피그레이션 가이드, Cisco IOS XE 17.x
- 9800 WLC용 체인을 생성하기 위한 인증서 정보 이해

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.