ISE 및 9800 WLC에서 Radius DTLS 구성

목차

소개

이 문서에서는 ISE와 9800 WLC 간의 RADIUS DTLS를 구성하는 데 필요한 인증서를 만드는 방법에 대해 설명합니다.

배경

RADIUS DTLS는 RADIUS 메시지가 DTLS(Data Transport Layer Security) 터널을 통해 전송되는 RADIUS 프로토콜의 보안 형식입니다. 인증 서버와 인증자 간에 이 터널을 생성하려면 인증서 집합이 필요합니다. 이 인증서 집합에서는 특정 EKU(Extended Key Usage) 인증서 확장을 설정해야 합니다. 특히 WLC 인증서에 대한 클라이언트 인증과 서버 인증 및 ISE 인증서에 대한 클라이언트 인증을 모두 설정해야 합니다.

사전 요구 사항

요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- 기본 작동을 위해 9800 WLC, 액세스 포인트(AP)를 구성하는 방법
- OpenSSL 애플리케이션 사용 방법
- PKI(Public Key Infrastructure) 및 디지털 인증서

사용되는 구성 요소

이 문서의 정보는 다음 소프트웨어 및 하드웨어 버전을 기반으로 합니다.

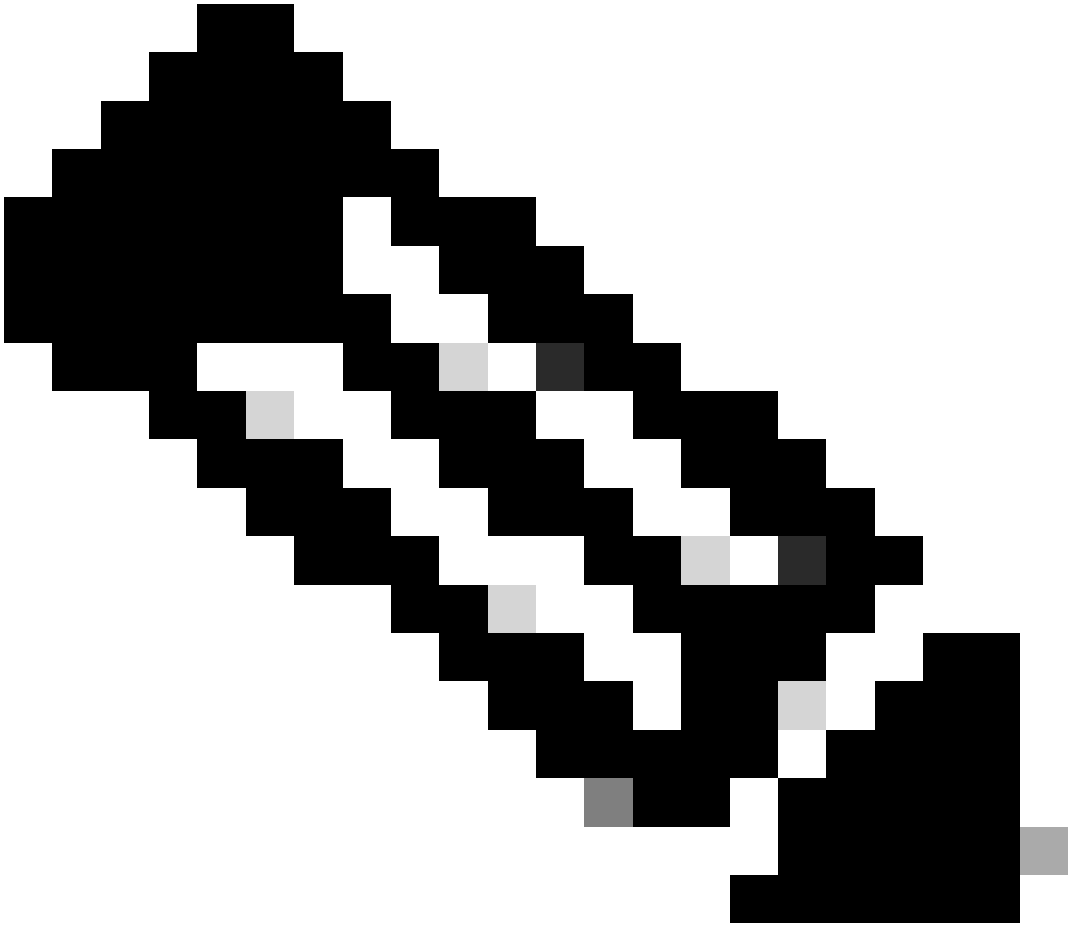
- OpenSSL 애플리케이션(버전 3.0.2).
- ISE(버전 3.1.0.518)
- 9800 WLC(버전 17.12.3)

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 CA 및 중간 CA를 사용하여 엔드포인트 인증서에 서명하는 2단계 인증 기관을 만드는 것이 목적입니다. 인증서가 서명되면 WLC 및 ISE로 가져옵니다. 마지막으로, 디바이스는 해당 인증서로 RADIUS DTLS 인증을 수행하도록 구성됩니다.



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.

선택 사항 - WLC 및 ISE RADIUS DTLS 디바이스 인증서 생성

RADIUS DTLS 프로토콜은 DTLS 터널을 생성하기 위해 ISE와 WLC 간에 인증서를 교환해야 합니다. 아직 유효한 인증서가 없는 경우 로컬 CA를 생성하여 인증서를 생성할 수 있습니다. Configure a Multi-level Certificate Authority on OpenSSL to Generate CIsco IOS® XE Compatible Certificates를 참조하고 시작 단계부터 종료 단계까지 문서에 설명된 단계를 수행합니다 중간 CA 인증서를 생성합니다.

openssl.cnf 파일에 구성 섹션 추가

openssl.cnf 컨피그레이션 파일을 열고 그 하단에서 유효한 CSR(Certificate Sign Request)을 생성하는 데 사용되는 WLC 및 ISE 섹션을 복사하여 붙여넣습니다.

ISE_device_req_ext 및 WLC_device_req_ext 섹션 각각은 CSR에 포함할 SAN 목록을 가리킵니다.

```
#Section used for CSR generation, it points to the list of subject alternative names to add them to CSR
[ ISE_device_req_ext ]
subjectAltName = @ISE_alt_names

[ WLC_device_req_ext ]
subjectAltName = @WLC_alt_names

#DEFINE HERE SANS/IPs NEEDED for **ISE** device certificates
[ISE_alt_names]
DNS.1   = ISE.example.com
DNS.2   = ISE2.example.com

#DEFINE HERE SANS/IPs NEEDED for **WLC** device certificates
[WLC_alt_names]
DNS.1   = WLC.example.com
DNS.2   = WLC2.example.com
```

보안 조치로서 CA는 CSR에 존재하는 모든 SAN을 재정의하여 서명합니다. 따라서 권한이 없는 디바이스는 사용할 수 없는 이름에 대한 유효한 인증서를 받을 수 없습니다. 서명된 인증서에 SAN을 다시 추가하려면 subjectAltName 매개 변수를 사용하여 CSR 생성에 사용된 것과 동일한 목록 SAN을 가리킵니다.

ISE는 인증서에 serverAuth 및 clientAuth EKU가 모두 있어야 하지만 WLC는 clientAuth만 필요합니다. 이러한 인증서는 extendedKeyUsage 매개변수로 서명된 인증서에 추가됩니다.

openssl.cnf 파일 하단의 인증서 서명에 사용되는 섹션을 복사하여 붙여 넣습니다.

```
#This section contains the extensions used for the device certificate sign
[ ISE_cert ]
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
#EKU client and server is needed for RADIUS DTLS on ISE
extendedKeyUsage = serverAuth, clientAuth
subjectAltName = @ISE_alt_names

[ WLC_cert ]
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
#EKU client is needed for RADIUS DTLS on WLC
extendedKeyUsage = clientAuth
subjectAltName = @WLC_alt_names
```

WLC 장치 인증서 만들기

IntermCA.db.certs라는 중간 CA 인증서 폴더 내에 OpenSSL이 설치된 시스템에 WLC 인증서를 저장할 새 폴더를 만듭니다. 새 폴더를 WLC라고 합니다.

```
mkdir ./IntermCA/IntermCA.db.certs/WLC
```

openssl.cnf 파일의 [WLC_alt_names] 섹션에서 DNS 매개변수를 수정합니다. 원하는 값에 대해 제공된 예제 이름을 변경합니다. 다음 값은 WLC 인증서의 SAN 필드를 채웁니다.

```
[WLC_alt_names]
DNS.1   = WLC.example.com    <-----Change the values after the equals sign
DNS.2   = WLC2.example.com   <-----Change the values after the equals sign
```

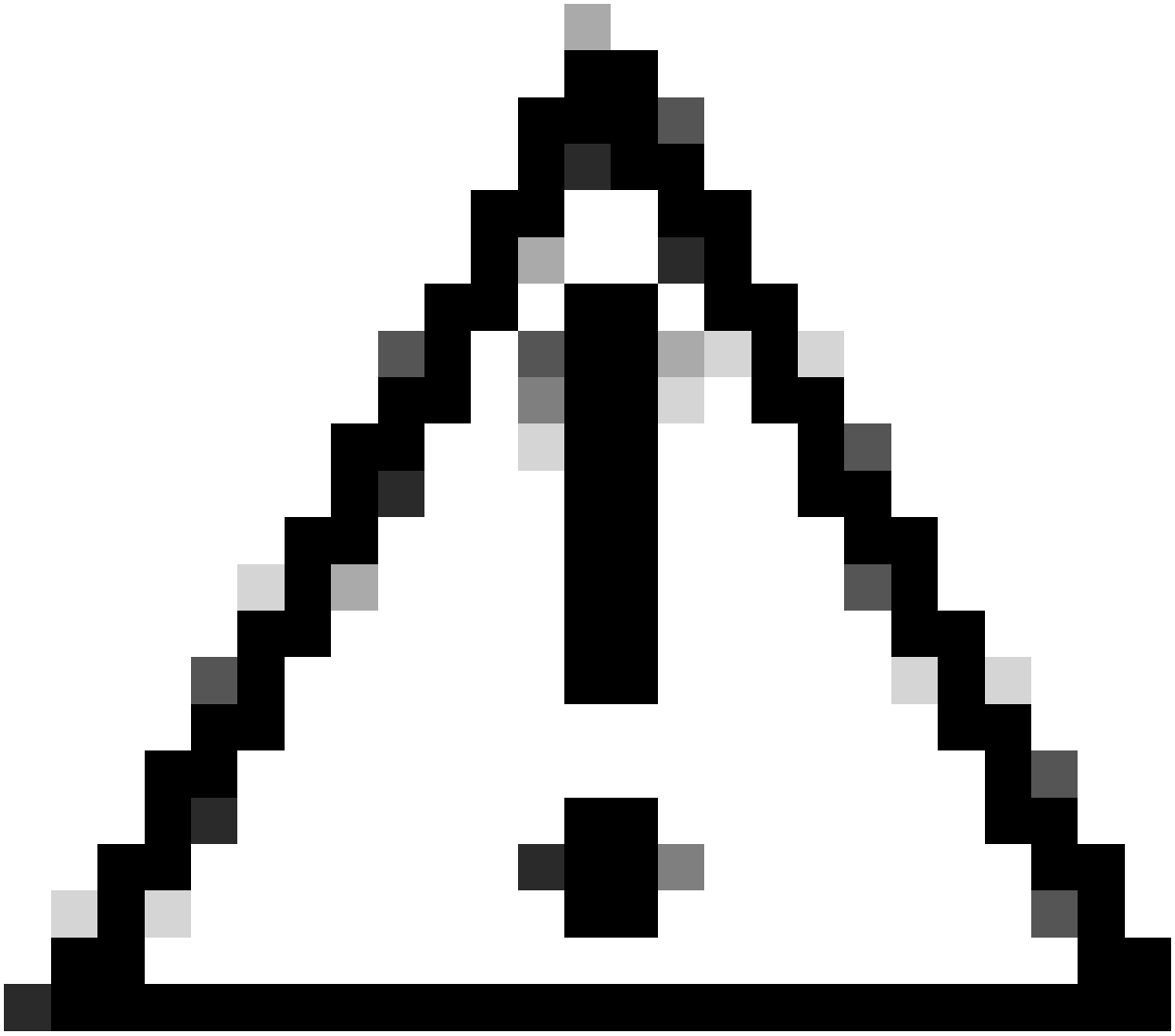
SAN용 WLC_device_req_ext 섹션의 정보를 사용하여 WLC 개인 키 및 WLC CSR을 생성합니다.

```
openssl req -newkey rsa:4096 -keyout ./IntermCA/IntermCA.db.certs/WLC/WLC.key -nodes -config openssl.cn
```

OpenSSL은 DN(Distinguished Name) 세부 정보를 입력하라는 대화형 프롬프트를 엽니다.

```
.........+..+.......+...........+...+......+.+............+..+++++++++++++++
++++++
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name [MX]:
State or province [CDMX]:
Locality [CDMX]:
Organization name [Cisco lab]:
Organizational unit [Cisco Wireless]:
Common name []:WLC.example.com
```

WLC 인증서 고유 이름 인터랙티브 프롬프트



주의: 대화식 프롬프트에서 제공하는 일반 이름(CN)은 openssl.cnf 파일의 [WLC_alt_names] 섹션에 있는 이름 중 하나와 동일해야 합니다.

IntermCA라는 CA를 사용하여 [WLC_cert]에 정의된 확장명을 사용하여 WLC.csr이라는 WLC CSR에 서명하고, 서명된 인증서를 ./IntermCA/IntermCA.db.certs/WLC 내부에 저장합니다. WLC 장치 인증서는 WLC.crt라고 합니다.

```
openssl ca -config openssl.cnf -extensions WLC_cert -name IntermCA -out ./IntermCA/IntermCA.db.certs/WL
```

9800 WLC를 가져오려면 인증서가 pfx 형식이어야 합니다. WLC 인증서에 서명한 CA의 체인을 포함하는 새 파일을 만듭니다. 이를 certfile이라고 합니다.

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

.pfx 파일을 만들려면 WLC 버전에 따라 다음 명령 중 하나를 실행합니다.

17.12.1 이전 버전의 경우:

```
openssl pkcs12 -export -macalg sha1 -legacy -descert -out ./IntermCA/IntermCA.db.certs/WLC/WLC.pfx -ink
```

버전 17.12.1 이상의 경우:

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

ISE 디바이스 인증서 생성

IntermCA.db.certs라는 중간 CA 인증서 폴더 내에 OpenSSL이 설치된 시스템에 ISE 인증서를 저장할 새 폴더를 만듭니다. 새 폴더를 ISE라고 합니다.

```
mkdir ./IntermCA/IntermCA.db.certs/ISE
```

openssl.cnf 파일의 [ISE_alt_names] 섹션에서 DNS 매개변수를 수정합니다. 원하는 값에 대해 제공된 예제 이름을 변경합니다. 이 값은 WLC 인증서의 SAN 필드에 채워집니다.

```
[ISE_alt_names]
DNS.1   = ISE.example.com   <-----Change the values after the equals sign
DNS.2   = ISE2.example.com  <-----Change the values after the equals sign
```

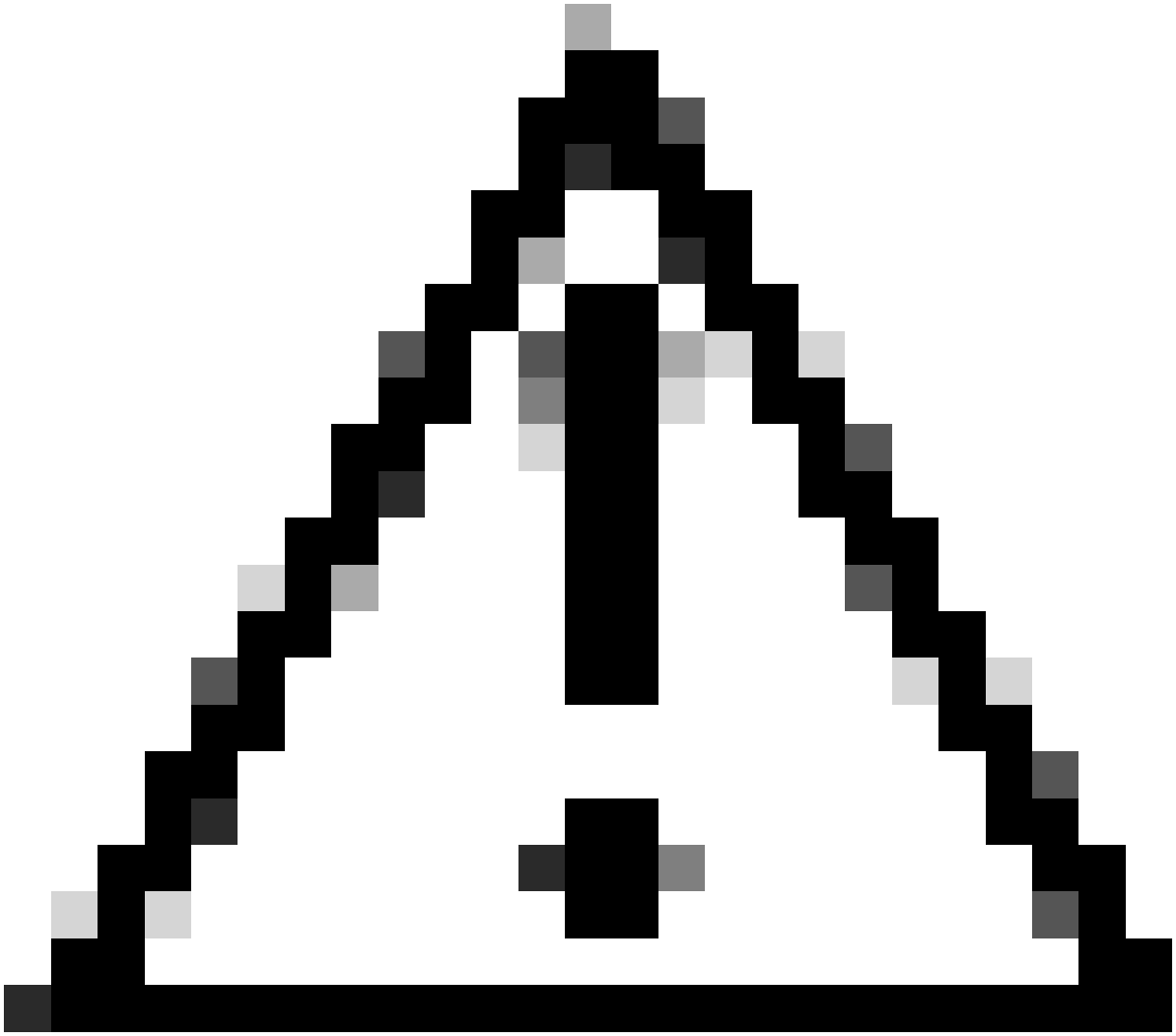
SAN용 ISE_device_req_ext 섹션의 정보를 사용하여 ISE 개인 키 및 ISE CSR을 생성합니다.

```
openssl req -newkey rsa:2048 -sha256 -keyout ./IntermCA/IntermCA.db.certs/ISE/ISE.key -nodes -config op
```

OpenSSL은 DN(Distinguished Name) 세부 정보를 입력하라는 대화형 프롬프트를 엽니다.

```
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name [MX]:
State or province [CDMX]:
Locality [CDMX]:
Organization name [Cisco lab]:
Organizational unit [Cisco Wireless]:
Common name []:ISE.example.com
```

ISE 인증서 고유 이름 인터랙티브 프롬프트



주의: 대화식 프롬프트에서 제공하는 CN은 openssl.cnf 파일의 [ISE_alt_names] 섹션에 있는 Names 중 하나와 정확히 동일해야 합니다.

IntermCA라는 CA를 사용하여 [ISE_cert]에 정의된 확장명을 사용하여 ISE.csr이라는 ISE CSR에 서명하고, 서명된 인증서를 ./IntermCA/IntermCA.db.certs/WLC 내부에 저장합니다. ISE 디바이스 인증서는 ISE.crt라고 합니다.

```
openssl ca -config openssl.cnf -extensions ISE_cert -name IntermCA -out ./IntermCA/IntermCA.db.certs/IS
```

디바이스에 인증서 가져오기

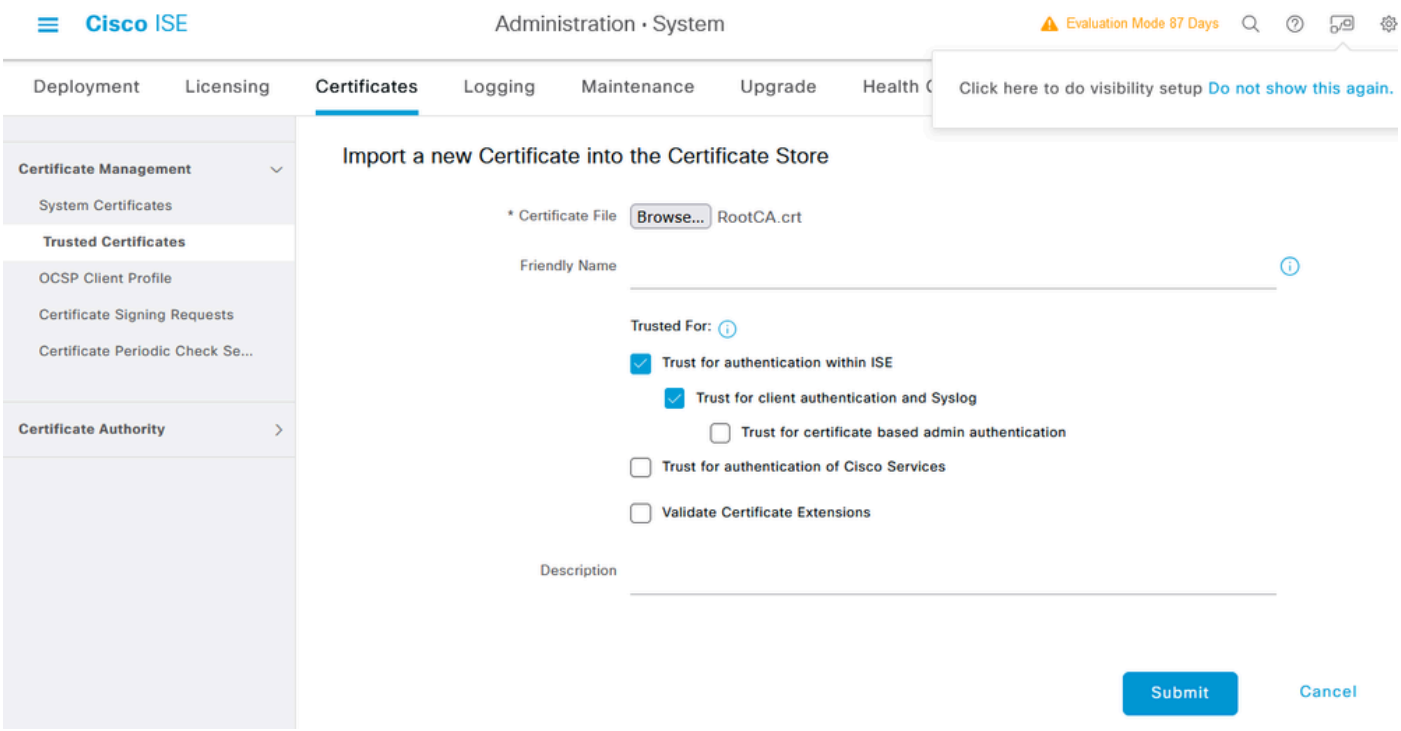
ISE로 인증서 가져오기

1. ISE 인증서 체인의 루트 CA 인증서를 신뢰할 수 있는 인증서 저장소로 가져옵니다.

2. Administration(관리)>System(시스템)>Certificates(인증서)>Trusted certificates(신뢰할 수 있는 인증서)로 이동합니다.

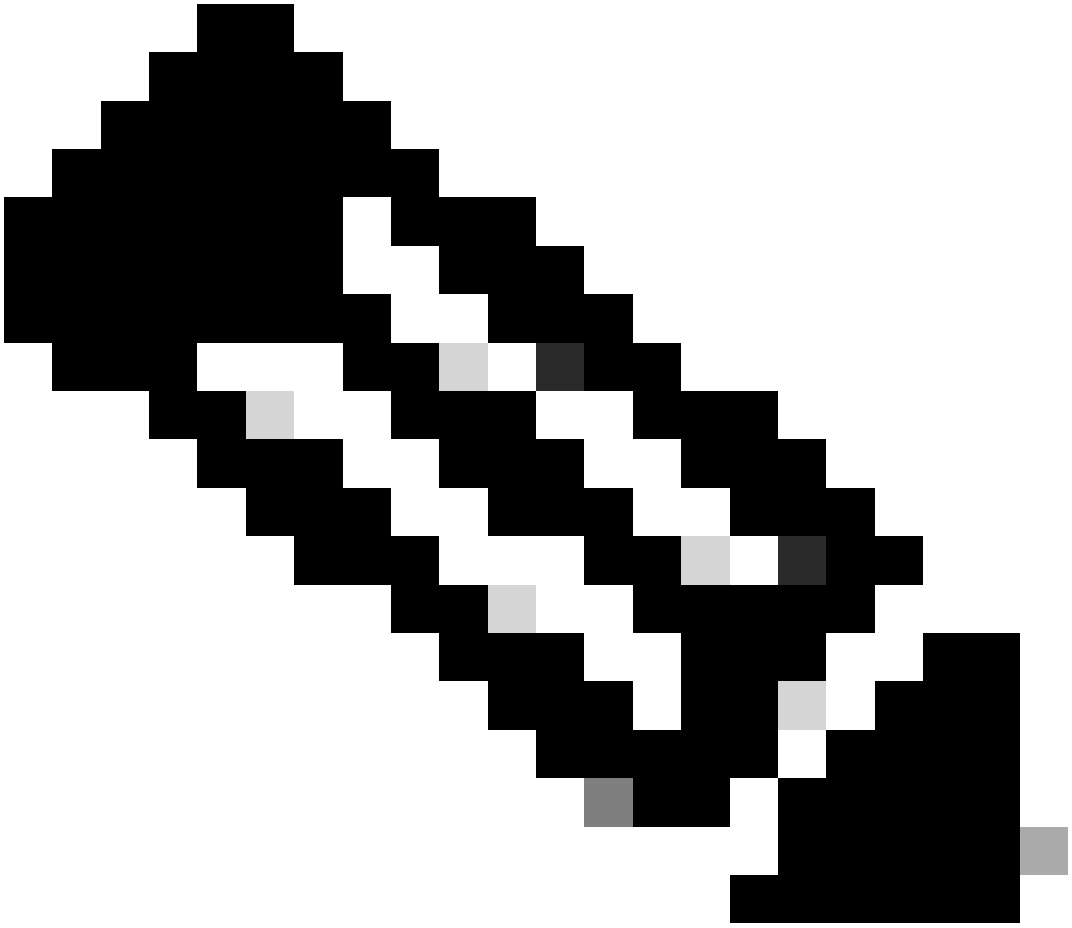
3. 찾아보기를 클릭하고 Root.crt 파일을 선택합니다.

4. Trust for authentication within ISE(ISE 내 인증 신뢰)와 Trust for client authentication and Syslog(클라이언트 인증 및 Syslog 신뢰) 확인란을 선택한 다음 Submit(제출)을 클릭합니다.

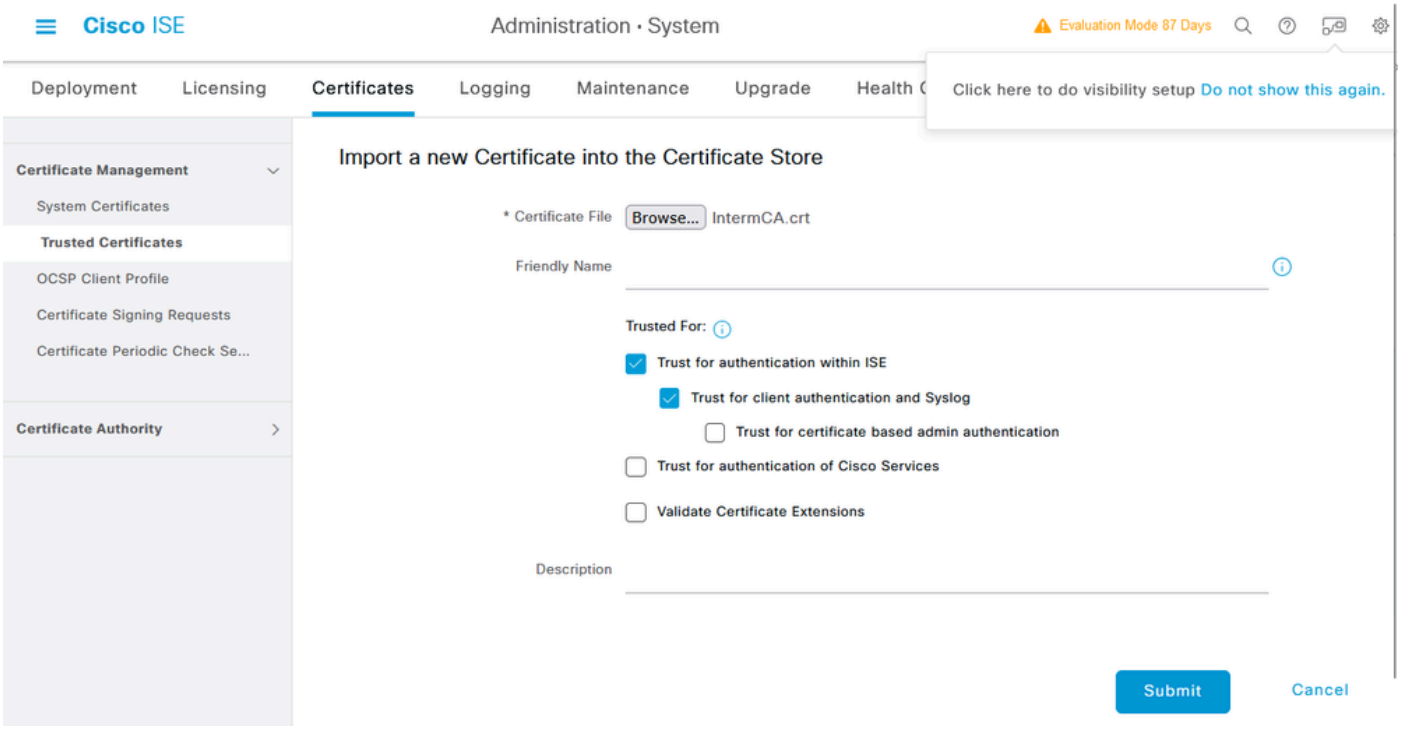


ISE 루트 CA 인증서 가져오기 대화 상자

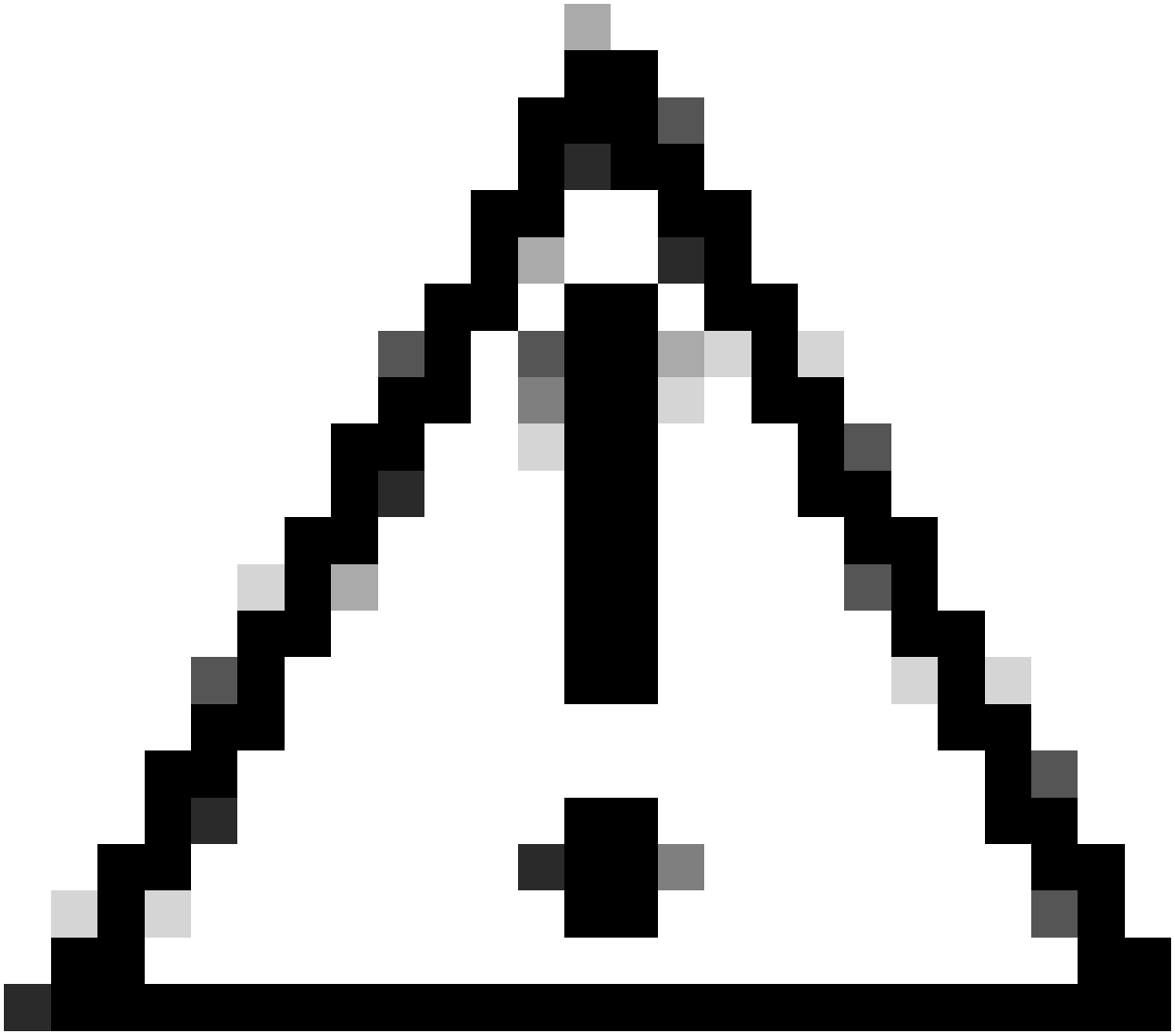
중간 인증서가 있는 경우 해당 인증서에 대해 동일한 작업을 수행합니다.



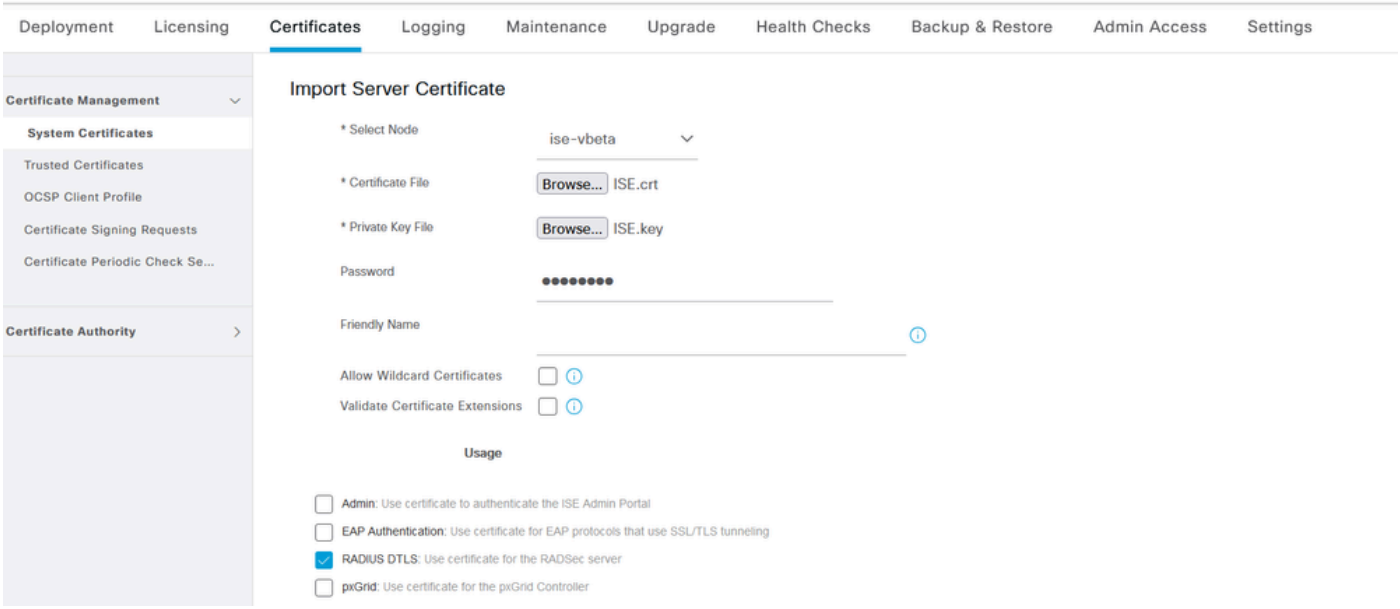
참고: ISE 인증서 검증 체인의 일부인 CA 인증서에 대해 단계를 반복합니다. 항상 루트 CA 인증서로 시작하고 체인의 가장 낮은 중간 CA 인증서로 마칩니다.



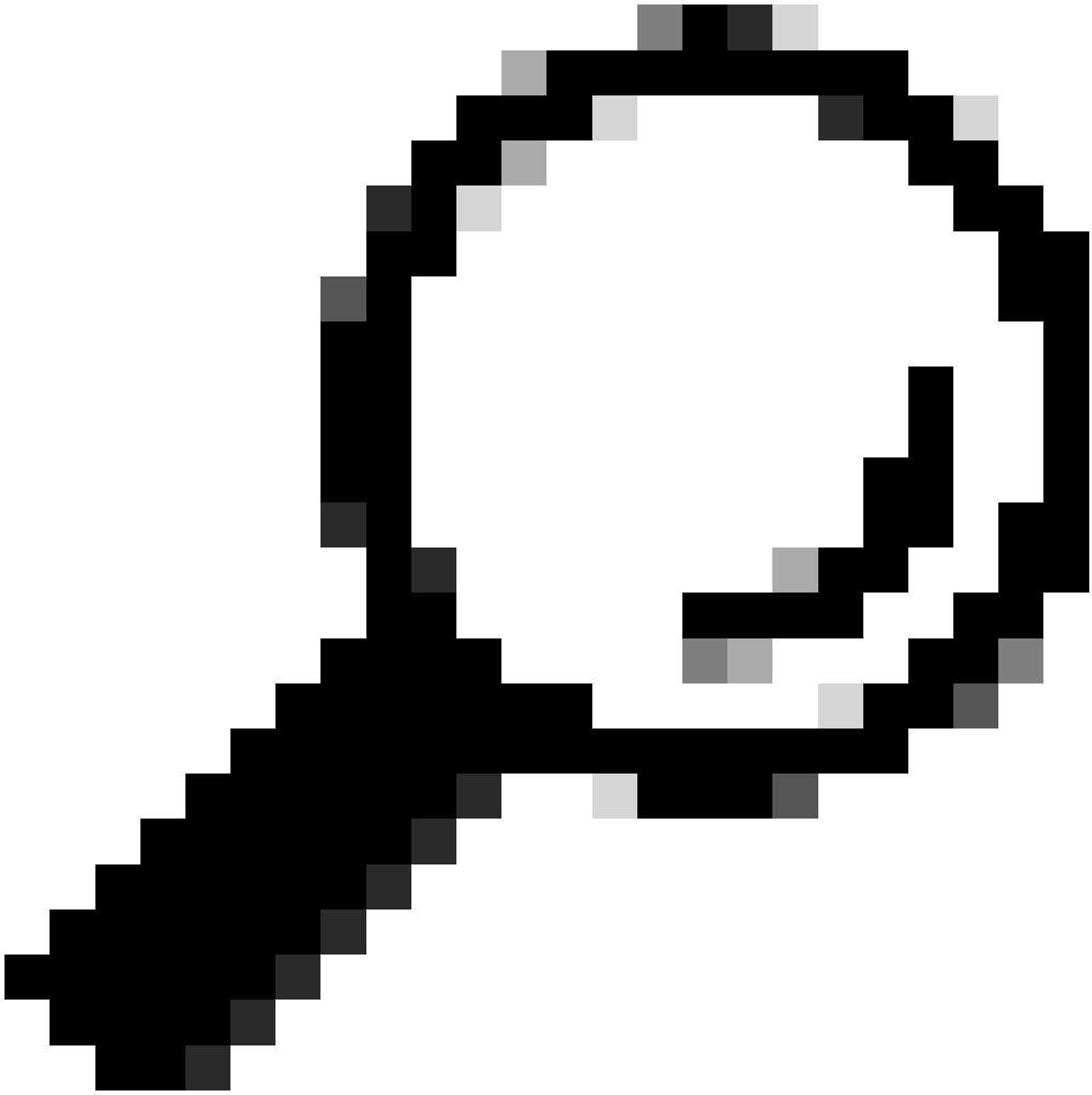
ISE 중간 CA 인증서 가져오기 대화 상자



주의: ISE 인증서 및 WLC 인증서가 서로 다른 CA에서 발급된 경우 WLC 인증서 체인에 속하는 모든 CA 인증서도 가져와야 합니다. ISE는 DTLS 인증서 교환에서 CA 인증서를 가져올 때까지 WLC 인증서를 승인하지 않습니다.



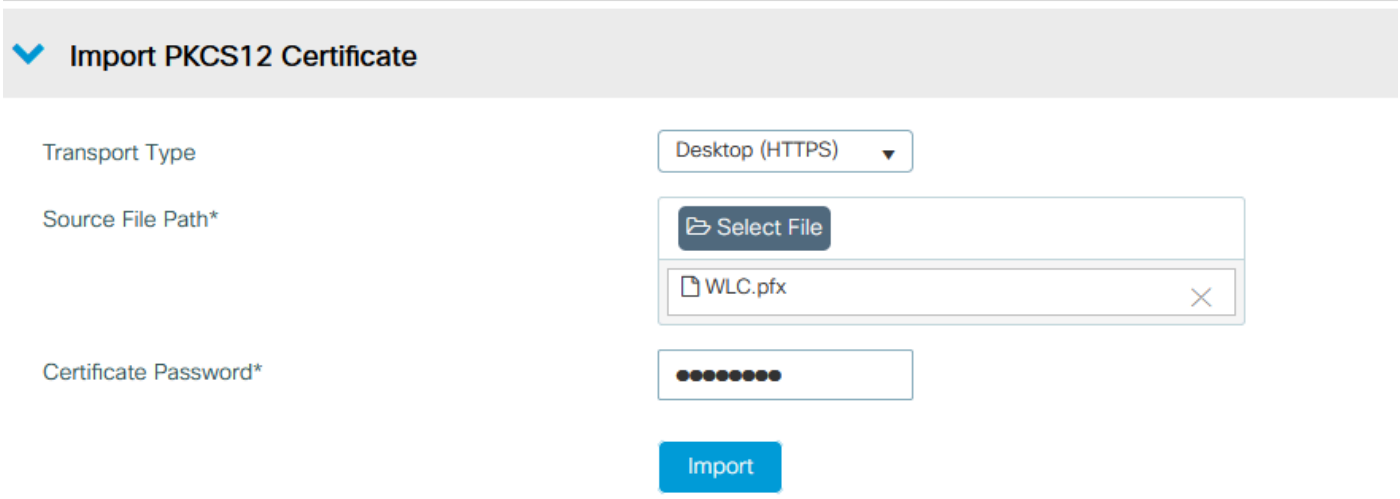
ISE 디바이스 인증서 가져오기 메뉴



팁: 이 단계에서는 ISE 디바이스 인증서만 가져오면 됩니다. 이 인증서는 ISE에서 DTLS 터널을 설정하기 위해 교환하는 인증서입니다. 이전에 가져온 CA 인증서를 사용하여 WLC 인증서가 확인되므로 WLC 디바이스 인증서 및 개인 키를 가져올 필요가 없습니다.

WLC에 인증서 가져오기

1. WLC에서 Configuration(컨피그레이션) > Security(보안) > PKI Management(PKI 관리)로 이동하고 Add Certificate(인증서 추가) 탭으로 이동합니다.
2. Import PKCS12 Certificate(PKCS12 인증서 가져오기) 드롭다운을 클릭하고 전송 유형을 Desktop(HTTPS)으로 설정합니다.
3. Select File(파일 선택) 버튼을 클릭하고 이전에 준비한 .pfx 파일을 선택합니다.
4. 가져오기 암호를 입력하고 마지막으로 Import(가져오기)를 클릭합니다.

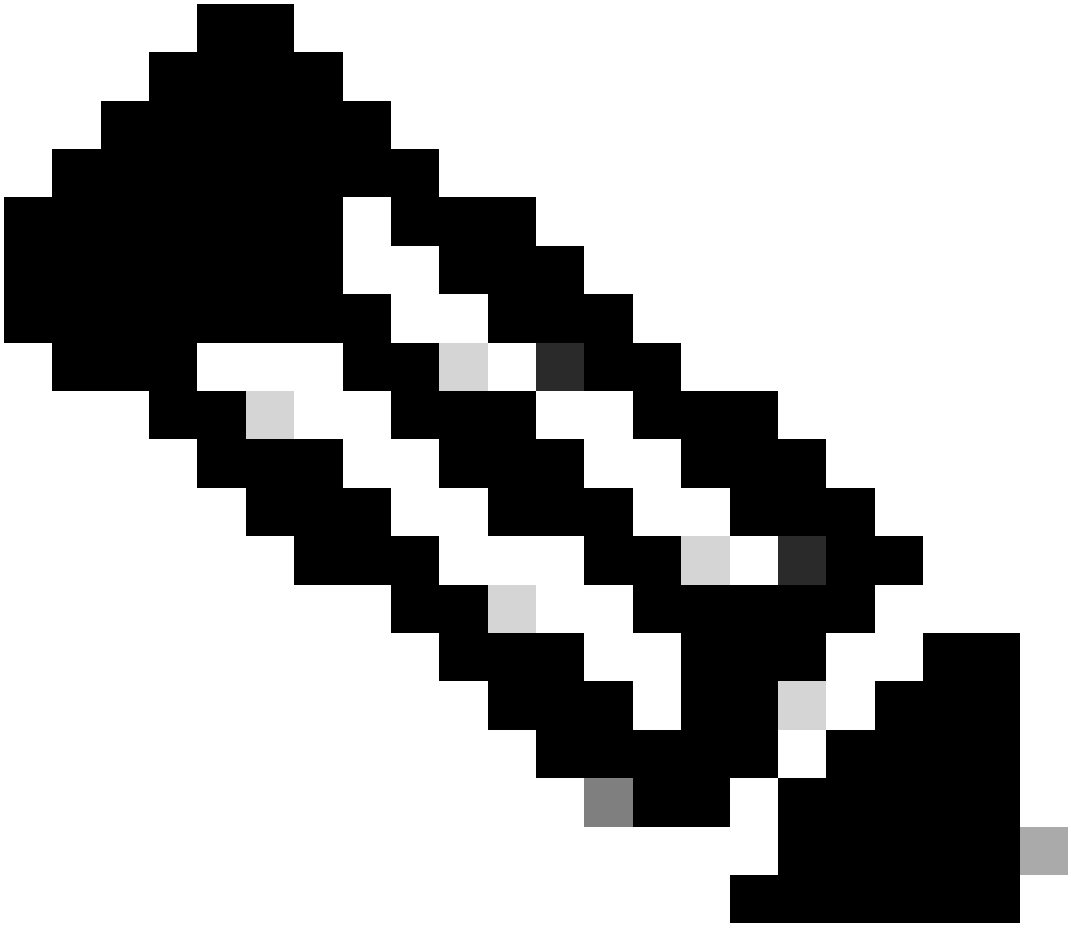


WLC 인증서 가져오기 대화 상자

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

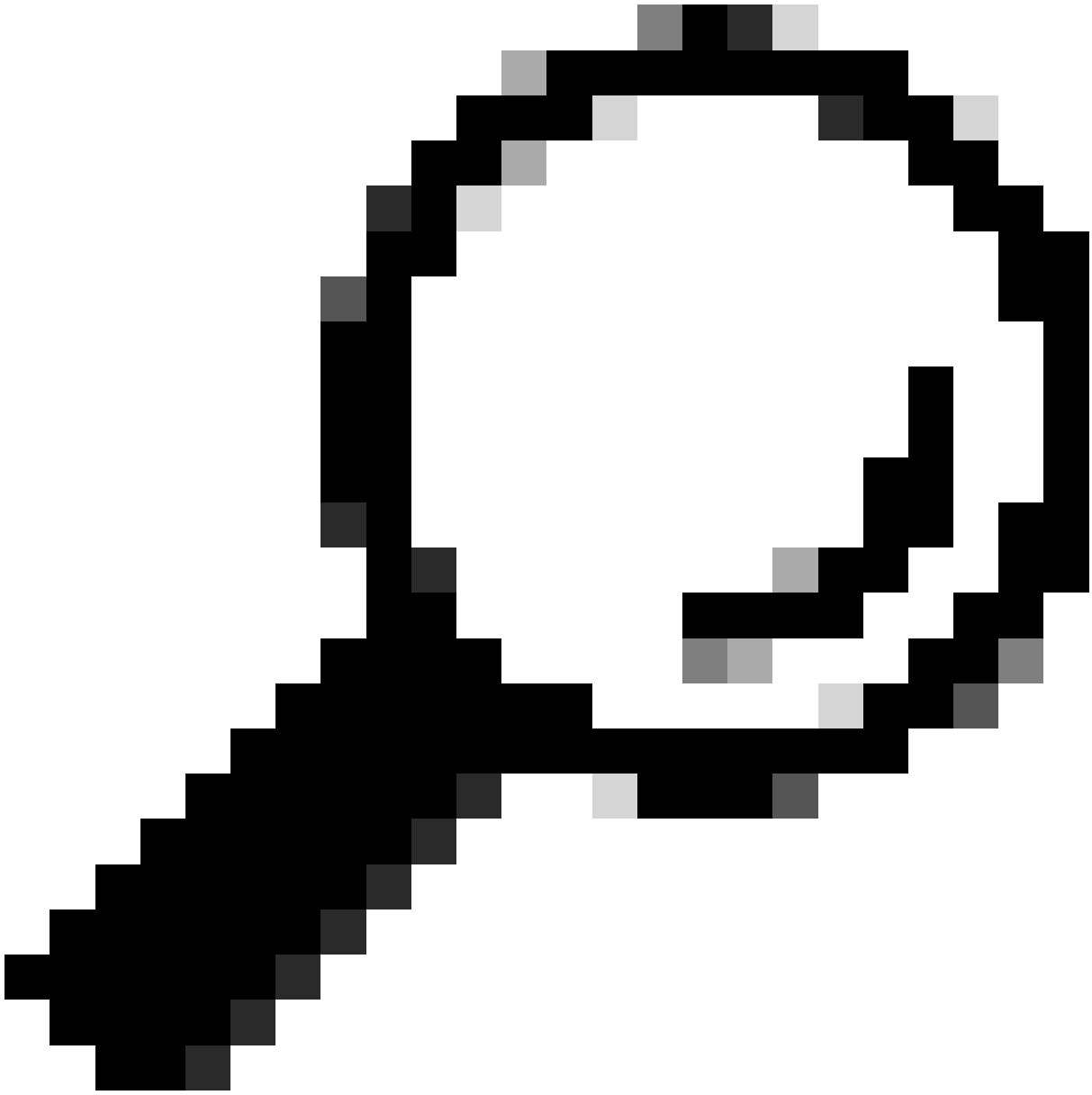
WLC에 네트워크를 통해 확인할 수 있는 인증서 해지 목록이 없는 경우 자동으로 생성된 각 신뢰 지점 내에서 해지 검사를 비활성화합니다.

```
9800#configure terminal

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

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



참고: Configure Multi-level CA on OpenSSL to Generate Cisco IOS XE Certificates(Cisco IOS XE 인증서를 생성하기 위해 OpenSSL에 다중 레벨 CA 구성) 문서를 사용하여 OpenSSL에 다중 레벨 CA를 생성한 경우 CRL 서버가 생성되지 않으므로 폐기 검사를 비활성화해야 합니다.

자동화된 가져오기는 WLC 인증서 및 해당 CA 인증서를 포함하는 데 필요한 신뢰 지점을 생성합니다.



팁: WLC 인증서가 ISE 인증서와 동일한 CA에서 발급된 경우 WLC 인증서 가져오기에서 자동으로 생성된 동일한 신뢰 지점을 사용할 수 있습니다. ISE 인증서를 별도로 가져올 필요가 없습니다.

WLC 인증서가 ISE 인증서와 다른 CA에서 발급된 경우 WLC가 ISE 디바이스 인증서를 신뢰하도록 WLC에 ISE CA 인증서도 가져와야 합니다.
루트 CA에 대한 새 신뢰 지점을 생성하고 ISE 루트 CA를 가져옵니다.

```
9800(config)#crypto pki trustpoint ISEroot
9800(ca-trustpoint)#revocation-check none
9800(ca-trustpoint)#enrollment terminal
9800(ca-trustpoint)#chain-validation stop
```

```
9800(ca-trustpoint)#exit
9800(config)#crypto pki authenticate ISEroot

Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself

------Paste the ISE root CA-----
```

ISE CA 체인의 다음 중간 CA 인증서, 즉 루트 CA에서 발급한 CA 인증서를 가져옵니다.

```
hamariomed1(config)#crypto pki trustpoint ISEintermediate
hamariomed1(ca-trustpoint)#revocation-check none
hamariomed1(ca-trustpoint)#chain-validation continue ISErootCA
hamariomed1(ca-trustpoint)#enrollment terminal
hamariomed1(ca-trustpoint)#exit

hamariomed1(config)#crypto pki authenticate ISEintermediate

Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself

------Paste the ISE intermediate CA-------
```

체인의 각 추가 CA에는 별도의 신뢰 지점이 필요합니다. 체인의 각 신뢰 지점은 chain-validation continue <Issuer trustpoint name> 명령을 사용하여 가져올 인증서의 발급자 인증서가 포함된 신뢰 지점을 참조해야 합니다.

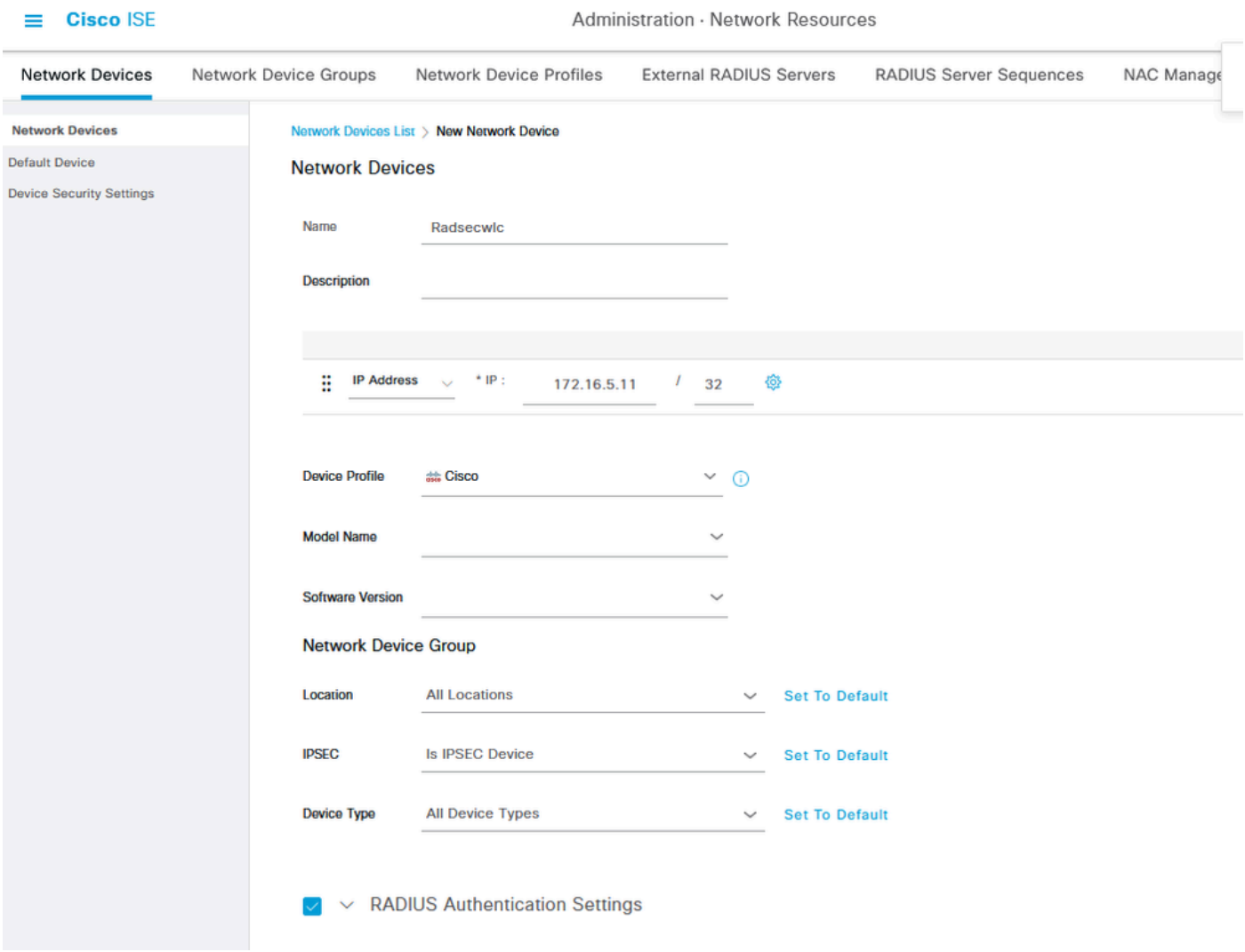
CA 체인에 포함된 수만큼의 CA 인증서를 가져옵니다. ISE 디바이스 인증서의 발급자 CA를 가져온 후 작업을 마쳤습니다. 이 신뢰 지점의 이름을 기록해 두십시오.

RADIUS DTLS가 작동하도록 WLC에서 ISE 디바이스 인증서를 가져올 필요가 없습니다.

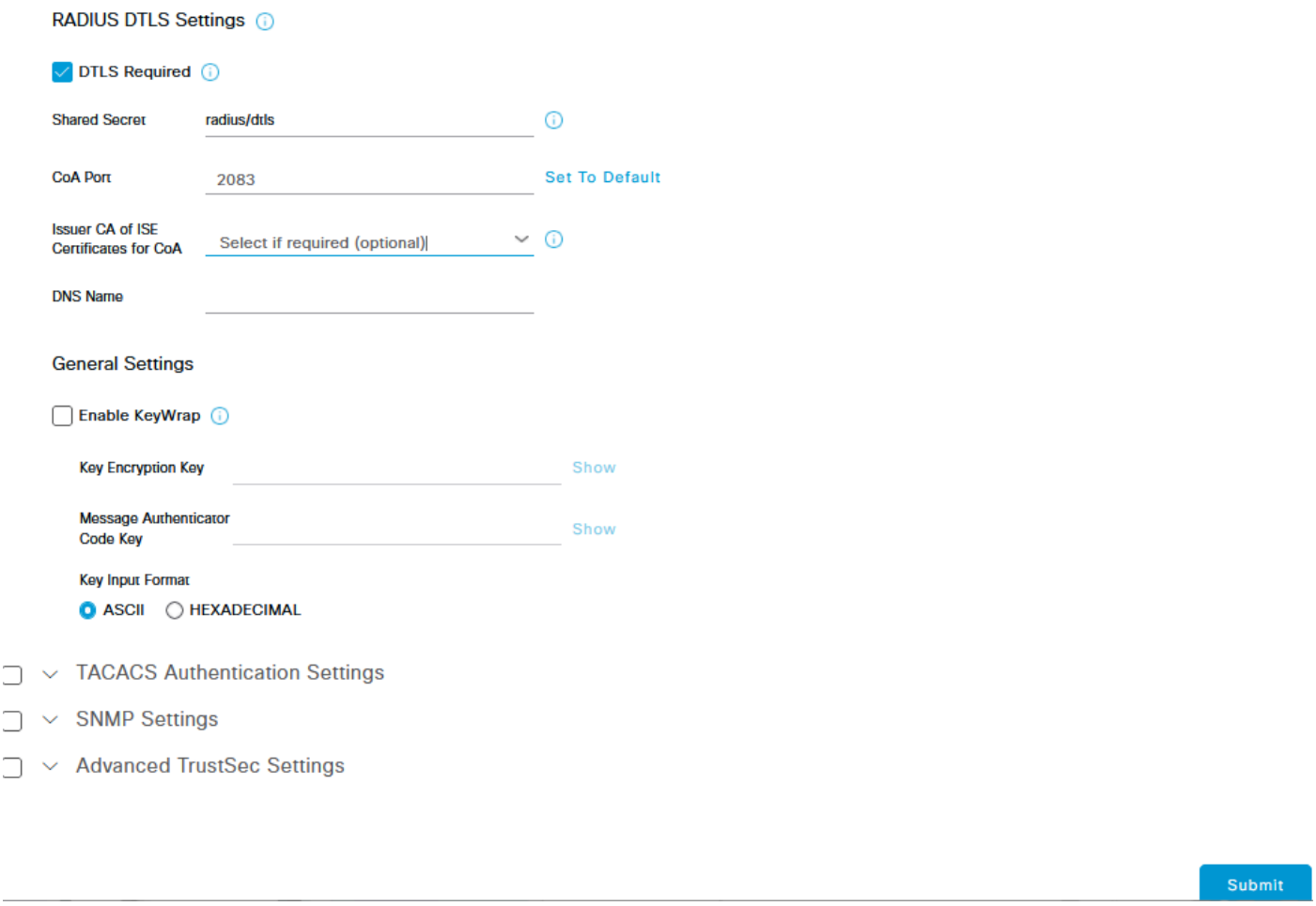
RADIUS DTLS 구성

ISE 구성

WLC를 네트워크 디바이스로 ISE에 추가합니다. 이렇게 하려면 Administration(관리)>Network Resources(네트워크 리소스)>Network devices(네트워크 디바이스)>Add(추가)로 이동합니다. 디바이스의 이름과 RADIUS 트래픽을 소싱하는 WLC 인터페이스의 IP를 입력합니다. 일반적으로 무선 관리 인터페이스 IP입니다. 아래로 스크롤하여 RADIUS Authentication Settings(RADIUS 인증 설정) 및 DTLS Required(DTLS 필수)를 선택하고 Submit(제출)을 클릭합니다.



새 네트워크 디바이스 컨피그레이션



ISE의 네트워크 디바이스에 대한 RADIUS DTLS 설정

WLC 컨피그레이션

ISE IP 주소 및 Radius DTLS의 기본 포트와 함께 새 Radius 서버를 정의합니다. 이 컨피그레이션은 CLI에서만 사용할 수 있습니다.

```
9800#configure terminal
9800(config)#radius server ISE
9800(config-radius-server)#address ipv4
```

```
9800(config-radius-server)#dtls port 2083
```

RADIUS DTLS는 공유 암호 radius/dtls를 사용해야 합니다. 9800 WLC는 이 키 이외의 구성된 키를 무시합니다.

```
9800(config-radius-server)#key radius/dtls
```

DTLSdtls trustpoint client
터널에 대해 교환할 WLC 디바이스 인증서가 포함된 신뢰 지점을 구성하려면 명령을 사용합니다.
ISEdtls trustpoint server
디바이스 인증서에 대한 발급자 CA가 포함된 신뢰 지점을 구성하려면 명령을 사용합니다.

클라이언트 및 서버 신뢰 지점 이름은 WLC 및 ISE 인증서가 동일한 CA에서 발급된 경우에만 동일합니다.

```
9800(config-radius-server)#dtls trustpoint client WLC.pfx
9800(config-radius-server)#dtls trustpoint server WLC.pfx
```

ISE 인증서에 있는 SAN(주체 대체 이름) 중 하나를 확인하도록 WLC를 구성합니다. 이 컨피그레이션은 인증서의 SAN 필드에 있는 SAN 중 하나와 정확히 일치해야 합니다.

9800 WLC는 SAN 필드에 대해 정규식 기반 일치를 수행하지 않습니다. dtls match-server-identity hostname *.example.com 즉, SAN 필드에 *.example.com이 있는 와일드카드 인증서에 대한 명령이 올바르지만 SAN 필드에 www.example.com이 포함된 인증서에 대한 동일한 명령이 올바르지 않습니다.

WLC는 이름 서버에 대해 이 이름을 확인하지 않습니다.

```
9800(config-radius-server)#dtls match-server-identity hostname ISE.example.com
9800(config-radius-server)#exit
```

인증에 새 Radius DTLS를 사용하려면 새 서버 그룹을 만듭니다.

```
9800(config)#aaa group server radius Radsec
9800(config-sg-radius)#server name ISE
9800(config-sg-radius)#exit
```

이 시점부터 WLC에서 다른 서버 그룹을 사용할 때 이 서버 그룹을 사용할 수 있습니다. 무선 클라이언트 인증에 이 서버를 사용하려면 Catalyst 9800 Wireless Controller Series에서 802.1X 인증 구성을 참조하십시오.

다음을 확인합니다.

인증서 정보 확인

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

```
openssl x509 -in

      -text -noout
```

전체 인증서 정보를 표시합니다. 이는 지정된 인증서의 발급자 CA를 확인하거나 인증서에 필요한 EKU 및 SAN이 포함되어 있는지 확인하는 데 유용합니다.

```
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)

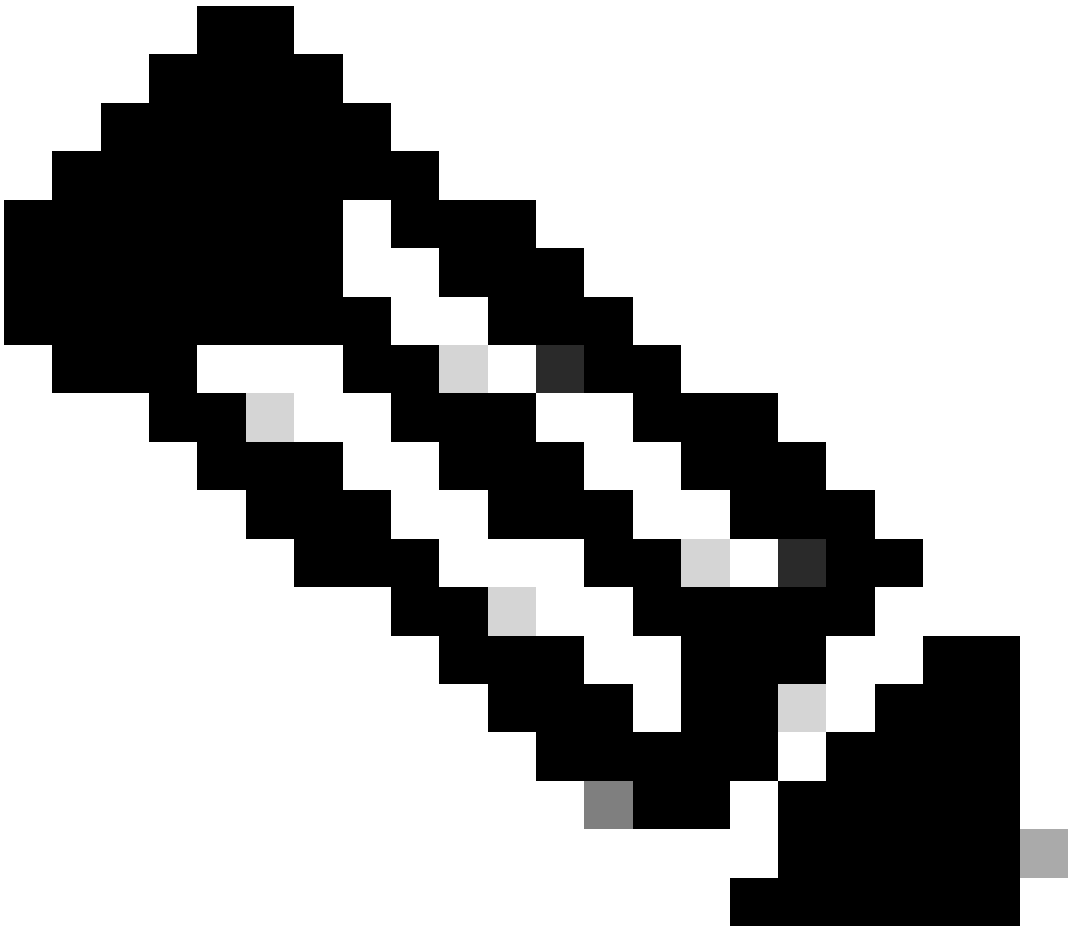
테스트 인증 수행

WLC에서 명령을 사용하여 Radius DTLS 기능을 테스트할 수 있습니다 test aaa group

new-code

```
9800#test aaa group Radsec testuser Cisco123 new-code
User successfully authenticated
```

```
USER ATTRIBUTES

username              0   "testuser"
```



> 참고: test 명령의 액세스 거부 출력은 WLC가 Access-Reject RADIUS 메시지를 받았음을 의미하며, 이 경우 RADIUS DTLS가 작동합니다. 그러나 DTLS 터널을 설정하지 못했음을 나타낼 수도 있습니다. test 명령은 두 시나리오를 모두 구분하지 않습니다. 문제 해결 섹션을 참조하여 문제가 있는지 확인하십시오.

문제 해결

실패한 인증의 원인을 검토하려면 테스트 인증을 수행하기 전에 이러한 명령을 활성화할 수 있습니다.

```
9800#debug radius
```

```
9800#debug radius radsec
9800#terminal monitor
```

디버그가 활성화된 성공적인 인증의 출력입니다.

```
9800#test aaa group Radsec testuser Cisco123 new-code
User successfully authenticated

USER ATTRIBUTES

username             0   "testuser"
9800#
Jul 18 21:24:38.301: %PARSER-5-HIDDEN: Warning!!! ' test platform-aaa group server-group Radsec user-na
Jul 18 21:24:38.313: RADIUS/ENCODE(00000000):Orig. component type = Invalid
Jul 18 21:24:38.313: RADIUS/ENCODE(00000000): dropping service type, "radius-server attribute 6 on-for-
Jul 18 21:24:38.313: RADIUS(00000000): Config NAS IP: 0.0.0.0
Jul 18 21:24:38.313: vrfid: [65535]  ipv6 tableid : [0]
Jul 18 21:24:38.313: idb is NULL
Jul 18 21:24:38.313: RADIUS(00000000): Config NAS IPv6: ::
Jul 18 21:24:38.313: RADIUS(00000000): sending
Jul 18 21:24:38.313: RADIUS/DECODE(00000000): There is no General DB. Want server details may not be sp
Jul 18 21:24:38.313: RADSEC: DTLS default secret
Jul 18 21:24:38.313: RADIUS/ENCODE: Best Local IP-Address 172.16.5.11 for Radius-Server 172.16.18.123
Jul 18 21:24:38.313: RADSEC: DTLS default secret
Jul 18 21:24:38.313: RADIUS(00000000): Send Access-Request to 172.16.18.123:2083 id 53808/10, len 54
RADIUS:  authenticator C3 4E 34 0A 91 EF 42 53 - 7E C8 BB 50 F3 98 B3 14
Jul 18 21:24:38.313: RADIUS:  User-Password       [2]   18  *
Jul 18 21:24:38.313: RADIUS:  User-Name           [1]   10  "testuser"
Jul 18 21:24:38.313: RADIUS:  NAS-IP-Address      [4]   6   172.16.5.11
Jul 18 21:24:38.313: RADIUS_RADSEC_ENQ_WAIT_Q: Success Server(172.16.18.123)/Id(10)
Jul 18 21:24:38.313: RADIUS_RADSEC_CLIENT_PROCESS: Got DATA SEND MSG
Jul 18 21:24:38.313: RADIUS_RADSEC_SOCK_SET:  0 Success
Jul 18 21:24:38.313: RADIUS_RADSEC_GENERATE_HASHKEY: hash key(0) generated for sock(0)
Jul 18 21:24:38.313: RADIUS_RADSEC_GENERATE_HASHBUCKET: hash bucket(0) generated for sock(0)
Jul 18 21:24:38.313: RADIUS_RADSEC_HASH_KEY_ADD_CTX: add [radius_radsec ctx(0x7522CE91BAC0)] succeedd f
Jul 18 21:24:38.313: RADIUS_RADSEC_GET_SOURCE_ADDR: Success
Jul 18 21:24:38.313: RADIUS_RADSEC_GET_SOCK_ADDR: Success
Jul 18 21:24:38.313: RADIUS_RADSEC_SET_LOCAL_SOCK: Success
Jul 18 21:24:38.313: RADIUS_RADSEC_SOCK_SET: Success
Jul 18 21:24:38.314: RADIUS_RADSEC_BIND_SOCKET: Success
Jul 18 21:24:38.314: RADIUS_RADSEC_CONN_SET_LPORT: Success
Jul 18 21:24:38.314: RADIUS_RADSEC_CONN_SET_SERVER_PORT: Success
Jul 18 21:24:38.314: RADIUS_RADSEC_CLIENT_HS_START: local port = 54509
Jul 18 21:24:38.314: RADIUS_RADSEC_SOCKET_CONNECT: Success
Jul 18 21:24:38.315: RADIUS_RADSEC_UPDATE_SVR_REF_CNT: Got radsec_data
Jul 18 21:24:38.315: RADIUS_RADSEC_UPDATE_SVR_REF_CNT: Got valid rctx, with server_handle B0000019
Jul 18 21:24:38.316: RADIUS_RADSEC_CLIENT_HS_START: TLS handshake in progress...(172.16.18.123/2083)
Jul 18 21:24:38.316: RADIUS_RADSEC_START_CONN_TIMER: Started (172.16.18.123/2083) for 5 secs
Jul 18 21:24:38.316: RADIUS_RADSEC_CONN_STATE_UPDATE: Success - State = 2
Jul 18 21:24:38.318: RADIUS_RADSEC_CLIENT_PROCESS: Got Socket Event
Jul 18 21:24:38.318: RADIUS_RADSEC_GENERATE_HASHBUCKET: hash bucket(0) generated for sock(0)
Jul 18 21:24:38.318: RADIUS_RADSEC_GENERATE_HASHKEY: hash key(0) generated for sock(0)
Jul 18 21:24:38.318: RADIUS_RADSEC_HASH_KEY_MATCH: hashkey1(0) matches hashkey2(0) TRUE
Jul 18 21:24:38.318: RADIUS_RADSEC_HASH_KEY_GET_CTX: radius radsec  sock_ctx(0x7522CE91BAC0:0) get for
Jul 18 21:24:38.318: RADIUS_RADSEC_PROCESS_SOCK_EVENT: Handle socket event for TLS handshake(172.16.18.
Jul 18 21:24:38.318: RADIUS_RADSEC_STOP_TIMER: Stopped (172.16.18.123/2083)
Jul 18 21:24:38.318: RADIUS_RADSEC_START_CONN_TIMER: Started (172.16.18.123/2083) for 5 secs
Jul 18 21:24:38.318: RADIUS_RADSEC_HS_CONTINUE: TLS handshake in progress...(172.16.18.123/2083)
```

```
Jul 18 21:24:38.318: RADIUS_RADSEC_SOCK_TLS_EVENT_HANDLE: Success
Jul 18 21:24:38.318: RADIUS_RADSEC_CLIENT_PROCESS: Got Socket Event
Jul 18 21:24:38.327: RADIUS_RADSEC_CLIENT_PROCESS: Got Socket Event
Jul 18 21:24:38.327: RADIUS_RADSEC_GENERATE_HASHBUCKET: hash bucket(0) generated for sock(0)
Jul 18 21:24:38.327: RADIUS_RADSEC_GENERATE_HASHKEY: hash key(0) generated for sock(0)
Jul 18 21:24:38.327: RADIUS_RADSEC_HASH_KEY_MATCH: hashkey1(0) matches hashkey2(0) TRUE
Jul 18 21:24:38.327: RADIUS_RADSEC_HASH_KEY_GET_CTX: radius radsec  sock_ctx(0x7522CE91BAC0:0) get for
Jul 18 21:24:38.327: RADIUS_RADSEC_PROCESS_SOCK_EVENT: Handle socket event for TLS handshake(172.16.18.
Jul 18 21:24:38.327: RADIUS_RADSEC_STOP_TIMER: Stopped (172.16.18.123/2083)
Jul 18 21:24:38.391: RADIUS_RADSEC_START_CONN_TIMER: Started (172.16.18.123/2083) for 5 secs
Jul 18 21:24:38.391: RADIUS_RADSEC_HS_CONTINUE: TLS handshake in progress...(172.16.18.123/2083)
Jul 18 21:24:38.391: RADIUS_RADSEC_SOCK_TLS_EVENT_HANDLE: Success
Jul 18 21:24:38.391: RADIUS_RADSEC_CLIENT_PROCESS: Got Socket Event
Jul 18 21:24:38.397: RADIUS_RADSEC_CLIENT_PROCESS: Got Socket Event
Jul 18 21:24:38.397: RADIUS_RADSEC_GENERATE_HASHBUCKET: hash bucket(0) generated for sock(0)
Jul 18 21:24:38.397: RADIUS_RADSEC_GENERATE_HASHKEY: hash key(0) generated for sock(0)
Jul 18 21:24:38.397: RADIUS_RADSEC_HASH_KEY_MATCH: hashkey1(0) matches hashkey2(0) TRUE
Jul 18 21:24:38.397: RADIUS_RADSEC_HASH_KEY_GET_CTX: radius radsec  sock_ctx(0x7522CE91BAC0:0) get for
Jul 18 21:24:38.397: RADIUS_RADSEC_PROCESS_SOCK_EVENT: Handle socket event for TLS handshake(172.16.18.
Jul 18 21:24:38.397: RADIUS_RADSEC_STOP_TIMER: Stopped (172.16.18.123/2083)
Jul 18 21:24:38.397: RADIUS_RADSEC_HS_CONTINUE: TLS handshake success!(172.16.18.123/2083) <-------- TL
Jul 18 21:24:38.397: RADIUS_RADSEC_CONN_STATE_UPDATE: Success - State = 3
Jul 18 21:24:38.397: RADIUS_RADSEC_UPDATE_SVR_REF_CNT: Got radsec_data
Jul 18 21:24:38.397: RADIUS_RADSEC_UPDATE_SVR_REF_CNT: Got valid rctx, with server_handle B0000019
Jul 18 21:24:38.397: RADIUS-RADSEC-HS-SUCCESS: Negotiated Cipher is ECDHE-RSA-AES256-GCM-SHA384
Jul 18 21:24:38.397: RADIUS_RADSEC_START_DATA_SEND: RADSEC HS Done, Start data send (172.16.18.123/2083
Jul 18 21:24:38.397: RADIUS_RADSEC_UNQUEUE_WAIT_Q: Success Server(172.16.18.123)/Id(10)
Jul 18 21:24:38.397: RADIUS_RADSEC_MSG_SEND: RADSEC Write SUCCESS(id=10)
Jul 18 21:24:38.397: RADIUS(00000000): Started 5 sec timeout
Jul 18 21:24:38.397: RADIUS_RADSEC_UNQUEUE_WAIT_Q: Empty Server(172.16.18.123)/Id(-1)
Jul 18 21:24:38.397: RADIUS_RADSEC_START_DATA_SEND: no more data available
Jul 18 21:24:38.397: RADIUS_RADSEC_IDLE_TIMER: Started (172.16.18.123/2083)
Jul 18 21:24:38.397: RADIUS-RADSEC-HS-SUCCESS: Success
Jul 18 21:24:38.397: RADIUS_RADSEC_SOCK_TLS_EVENT_HANDLE: Success
Jul 18 21:24:38.397: RADIUS_RADSEC_CLIENT_PROCESS: Got Socket Event
Jul 18 21:24:38.453: RADIUS_RADSEC_CLIENT_PROCESS: Got Socket Event
Jul 18 21:24:38.453: RADIUS_RADSEC_GENERATE_HASHBUCKET: hash bucket(0) generated for sock(0)
Jul 18 21:24:38.453: RADIUS_RADSEC_GENERATE_HASHKEY: hash key(0) generated for sock(0)
Jul 18 21:24:38.453: RADIUS_RADSEC_HASH_KEY_MATCH: hashkey1(0) matches hashkey2(0) TRUE
Jul 18 21:24:38.453: RADIUS_RADSEC_HASH_KEY_GET_CTX: radius radsec  sock_ctx(0x7522CE91BAC0:0) get for
Jul 18 21:24:38.453: RADIUS_RADSEC_MSG_RECV: RADSEC Bytes read= 20, Err= 0
Jul 18 21:24:38.453: RADIUS_RADSEC_SOCK_READ_EVENT_HANDLE: Radius length is 113
Jul 18 21:24:38.453: RADIUS_RADSEC_SOCK_READ_EVENT_HANDLE: Going to read rest 93 bytes
Jul 18 21:24:38.453: RADIUS_RADSEC_MSG_RECV: RADSEC Bytes read= 93, Err= 0
Jul 18 21:24:38.453: RADIUS_RADSEC_SOCK_READ_EVENT_HANDLE: linktype = 7 - src port = 2083 - dest port =
Jul 18 21:24:38.453: RADIUS: Received from id 54509/10 172.16.18.123:2083, Access-Accept, len 113 <----
RADIUS:  authenticator 4E CE 96 63 41 4B 43 04 - C7 A2 B5 05 C2 78 A7 0D
Jul 18 21:24:38.453: RADIUS:  User-Name           [1]   10  "testuser"
Jul 18 21:24:38.453: RADIUS:  Class               [25]  83
RADIUS:   43 41 43 53 3A 61 63 31 30 31 32 37 62 64 38 74  [CACS:ac10127bd8t]
RADIUS:   47 58 50 47 4E 63 6C 57 76 2F 39 67 44 66 51 67  [GXPGNclWv/9gDfQg]
RADIUS:   63 4A 76 6C 35 47 72 33 71 71 47 36 4C 66 35 59  [cJvl5Gr3qqG6Lf5Y]
RADIUS:   52 42 2F 7A 57 55 39 59 3A 69 73 65 2D 76 62 65  [RB/zWU9Y:ise-vbe]
RADIUS:   74 61 6E 63 6F 2F 35 31 30 34 33 39 38 32 36 2F  [tanco/510439826/]
RADIUS:   39                  [ 9]
Jul 18 21:24:38.453: RADSEC: DTLS default secret
Jul 18 21:24:38.453: RADIUS/DECODE(00000000): There is no General DB. Reply server details may not be r
Jul 18 21:24:38.453: RADIUS(00000000): Received from id 54509/10
```

WLC에서 보고된 알 수 없는 CA

WLC가 ISE에서 제공하는 인증서를 검증할 수 없으면 DTLS 터널을 생성하지 못하고 인증이 실패합니다.
이 경우 표시되는 디버그 메시지의 예입니다.

```
9800#test aaa group Radsec testuser Cisco123 new-code

Jul 19 00:59:09.695: %PARSER-5-HIDDEN: Warning!!! ' test platform-aaa group server-group Radsec user-na
Jul 19 00:59:09.706: RADIUS/ENCODE(00000000):Orig. component type = Invalid
Jul 19 00:59:09.707: RADIUS/ENCODE(00000000): dropping service type, "radius-server attribute 6 on-for-
Jul 19 00:59:09.707: RADIUS(00000000): Config NAS IP: 0.0.0.0
Jul 19 00:59:09.707: vrfid: [65535]  ipv6 tableid : [0]
Jul 19 00:59:09.707: idb is NULL
Jul 19 00:59:09.707: RADIUS(00000000): Config NAS IPv6: ::
Jul 19 00:59:09.707: RADIUS(00000000): sending
Jul 19 00:59:09.707: RADIUS/DECODE(00000000): There is no General DB. Want server details may not be sp
Jul 19 00:59:09.707: RADSEC: DTLS default secret
Jul 19 00:59:09.707: RADIUS/ENCODE: Best Local IP-Address 172.16.5.11 for Radius-Server 172.16.18.123
Jul 19 00:59:09.707: RADSEC: DTLS default secret
Jul 19 00:59:09.707: RADIUS(00000000): Send Access-Request to 172.16.18.123:2083 id 52764/13, len 54
RADIUS:  authenticator E8 09 1D B0 72 50 17 E6 - B4 27 F6 E3 18 25 16 64
Jul 19 00:59:09.707: RADIUS:  User-Password       [2]   18  *
Jul 19 00:59:09.707: RADIUS:  User-Name           [1]   10  "testuser"
Jul 19 00:59:09.707: RADIUS:  NAS-IP-Address      [4]   6   172.16.5.11
Jul 19 00:59:09.707: RADIUS_RADSEC_ENQ_WAIT_Q: Success Server(172.16.18.123)/Id(13)
Jul 19 00:59:09.707: RADIUS_RADSEC_CLIENT_PROCESS: Got DATA SEND MSG
Jul 19 00:59:09.707: RADIUS_RADSEC_SOCK_SET:  0 Success
Jul 19 00:59:09.707: RADIUS_RADSEC_GENERATE_HASHKEY: hash key(0) generated for sock(0)
Jul 19 00:59:09.707: RADIUS_RADSEC_GENERATE_HASHBUCKET: hash bucket(0) generated for sock(0)
Jul 19 00:59:09.707: RADIUS_RADSEC_HASH_KEY_ADD_CTX: add [radius_radsec ctx(0x7522CE91BAC0)] succeedd f
Jul 19 00:59:09.707: RADIUS_RADSEC_GET_SOURCE_ADDR: Success
Jul 19 00:59:09.707: RADIUS_RADSEC_GET_SOCK_ADDR: Success
Jul 19 00:59:09.707: RADIUS_RADSEC_SET_LOCAL_SOCK: Success
Jul 19 00:59:09.707: RADIUS_RADSEC_SOCK_SET: Success
Jul 19 00:59:09.707: RADIUS_RADSEC_BIND_SOCKET: Success
Jul 19 00:59:09.707: RADIUS_RADSEC_CONN_SET_LPORT: Success
Jul 19 00:59:09.707: RADIUS_RADSEC_CONN_SET_SERVER_PORT: Success
Jul 19 00:59:09.707: RADIUS_RADSEC_CLIENT_HS_START: local port = 49556
Jul 19 00:59:09.707: RADIUS_RADSEC_SOCKET_CONNECT: Success
Jul 19 00:59:09.709: RADIUS_RADSEC_UPDATE_SVR_REF_CNT: Got radsec_data
Jul 19 00:59:09.709: RADIUS_RADSEC_UPDATE_SVR_REF_CNT: Got valid rctx, with server_handle B0000019
Jul 19 00:59:09.709: RADIUS_RADSEC_CLIENT_HS_START: TLS handshake in progress...(172.16.18.123/2083)
Jul 19 00:59:09.709: RADIUS_RADSEC_START_CONN_TIMER: Started (172.16.18.123/2083) for 5 secsUser reject

uwu-9800#
Jul 19 00:59:09.709: RADIUS_RADSEC_CONN_STATE_UPDATE: Success - State = 2
Jul 19 00:59:09.711: RADIUS_RADSEC_CLIENT_PROCESS: Got Socket Event
Jul 19 00:59:09.711: RADIUS_RADSEC_GENERATE_HASHBUCKET: hash bucket(0) generated for sock(0)
Jul 19 00:59:09.711: RADIUS_RADSEC_GENERATE_HASHKEY: hash key(0) generated for sock(0)
Jul 19 00:59:09.711: RADIUS_RADSEC_HASH_KEY_MATCH: hashkey1(0) matches hashkey2(0) TRUE
Jul 19 00:59:09.711: RADIUS_RADSEC_HASH_KEY_GET_CTX: radius radsec  sock_ctx(0x7522CE91BAC0:0) get for
Jul 19 00:59:09.711: RADIUS_RADSEC_PROCESS_SOCK_EVENT: Handle socket event for TLS handshake(172.16.18.
Jul 19 00:59:09.711: RADIUS_RADSEC_STOP_TIMER: Stopped (172.16.18.123/2083)
Jul 19 00:59:09.711: RADIUS_RADSEC_START_CONN_TIMER: Started (172.16.18.123/2083) for 5 secs
Jul 19 00:59:09.711: RADIUS_RADSEC_HS_CONTINUE: TLS handshake in progress...(172.16.18.123/2083)
Jul 19 00:59:09.711: RADIUS_RADSEC_SOCK_TLS_EVENT_HANDLE: Success
Jul 19 00:59:09.713: RADIUS_RADSEC_CLIENT_PROCESS: Got Socket Event
```

```
Jul 19 00:59:09.720: RADIUS_RADSEC_CLIENT_PROCESS: Got Socket Event
Jul 19 00:59:09.720: RADIUS_RADSEC_GENERATE_HASHBUCKET: hash bucket(0) generated for sock(0)
Jul 19 00:59:09.720: RADIUS_RADSEC_GENERATE_HASHKEY: hash key(0) generated for sock(0)
Jul 19 00:59:09.720: RADIUS_RADSEC_HASH_KEY_MATCH: hashkey1(0) matches hashkey2(0) TRUE
Jul 19 00:59:09.720: RADIUS_RADSEC_HASH_KEY_GET_CTX: radius radsec  sock_ctx(0x7522CE91BAC0:0) get for
Jul 19 00:59:09.720: RADIUS_RADSEC_PROCESS_SOCK_EVENT: Handle socket event for TLS handshake(172.16.18.
Jul 19 00:59:09.720: RADIUS_RADSEC_STOP_TIMER: Stopped (172.16.18.123/2083)
Jul 19 00:59:09.722: RADIUS_RADSEC_HS_CONTINUE: TLS handshake failed!
Jul 19 00:59:09.722: RADIUS_RADSEC_UNQUEUE_WAIT_Q: Success Server(172.16.18.123)/Id(13)
Jul 19 00:59:09.722: RADIUS_RADSEC_FAILOVER_HANDLER:Failng-over to new server = 0x0
Jul 19 00:59:09.722: RADIUS_RADSEC_UNQUEUE_WAIT_Q: Empty Server(172.16.18.123)/Id(-1)
Jul 19 00:59:09.722: RADIUS_RADSEC_FAILOVER_HANDLER: no more data available
Jul 19 00:59:09.722: RADIUS_RADSEC_STOP_TIMER: Stopped (172.16.18.123/2083)
Jul 19 00:59:09.722: RADIUS_RADSEC_CONN_CLOSE: Cleaned up timers for Radius RADSEC ctx
Jul 19 00:59:09.722: RADIUS_RADSEC_GENERATE_HASHKEY: hash key(0) generated for sock(0)
Jul 19 00:59:09.722: RADIUS_RADSEC_GENERATE_HASHBUCKET: hash bucket(0) generated for sock(0)
Jul 19 00:59:09.722: RADIUS_RADSEC_HASH_KEY_DEL_CTX: remove [radius_radsec  ctx(0x7522CE91BAC0)] succee
Jul 19 00:59:09.722: RADIUS_RADSEC_CONN_CLOSE: Hash table entry removed for RADSEC sock ctx
Jul 19 00:59:09.723: RADIUS_RADSEC_CONN_CLOSE: Success
Jul 19 00:59:09.723: RADIUS_RADSEC_SOCK_TLS_EVENT_HANDLE: Failed to complete TLS handshake <----------D
Jul 19 00:59:09.723: RADIUS_RADSEC_STOP_TIMER: Stopped (172.16.18.123/2083)
Jul 19 00:59:09.723: RADIUS_RADSEC_CONN_CLOSE: Cleaned up timers for Radius RADSEC ctx
Jul 19 00:59:09.723: RADIUS_RADSEC_GENERATE_HASHKEY: hash key(-1) generated for sock(-1)
Jul 19 00:59:09.723: RADIUS_RADSEC_GENERATE_HASHBUCKET: hash bucket(-1) generated for sock(-1)
uwu-9800#
Jul 19 00:59:09.723: RADIUS_RADSEC_HASH_KEY_DEL_CTX: remove [radius_radsec  ctx(0x7522CE91BAC0)] succee
Jul 19 00:59:09.723: RADIUS_RADSEC_CONN_CLOSE: Hash table entry removed for RADSEC sock ctx
Jul 19 00:59:09.723: RADIUS_RADSEC_CONN_CLOSE: Success
Jul 19 00:59:09.723: RADIUS_RADSEC_SOCK_TLS_EVENT_HANDLE: Error
Jul 19 00:59:09.723: RADIUS_RADSEC_PROCESS_SOCK_EVENT: failed to hanlde radsec hs event
Jul 19 00:59:09.723: RADIUS/DECODE: No response from radius-server; parse response; FAIL
Jul 19 00:59:09.723: RADIUS/DECODE: Case error(no response/ bad packet/ op decode);parse response; FAIL
Jul 19 00:59:09.723: RADIUS_RADSEC_CLIENT_PROCESS: Got Socket Event
Jul 19 00:59:10.718: %RADSEC_AUDIT_MESSAGE-3-FIPS_AUDIT_FCS_RADSEC_SERVER_CERTIFICATE_VALIDATION_FAILUR
Jul 19 00:59:10.718: %RADSEC_AUDIT_MESSAGE-3-FIPS_AUDIT_FCS_RADSEC_SERVER_IDENTITY_CHECK_FAILURE: Chass
Jul 19 00:59:10.718: %RADSEC_AUDIT_MESSAGE-6-FIPS_AUDIT_FCS_DTLSC_DTLS_SESSION_CLOSED: Chassis 1 R0/0:
```

이를 수정하려면 WLC에 구성된 ID가 ISE 인증서에 포함된 SAN 중 하나와 정확하게 일치하는지 확인합니다.

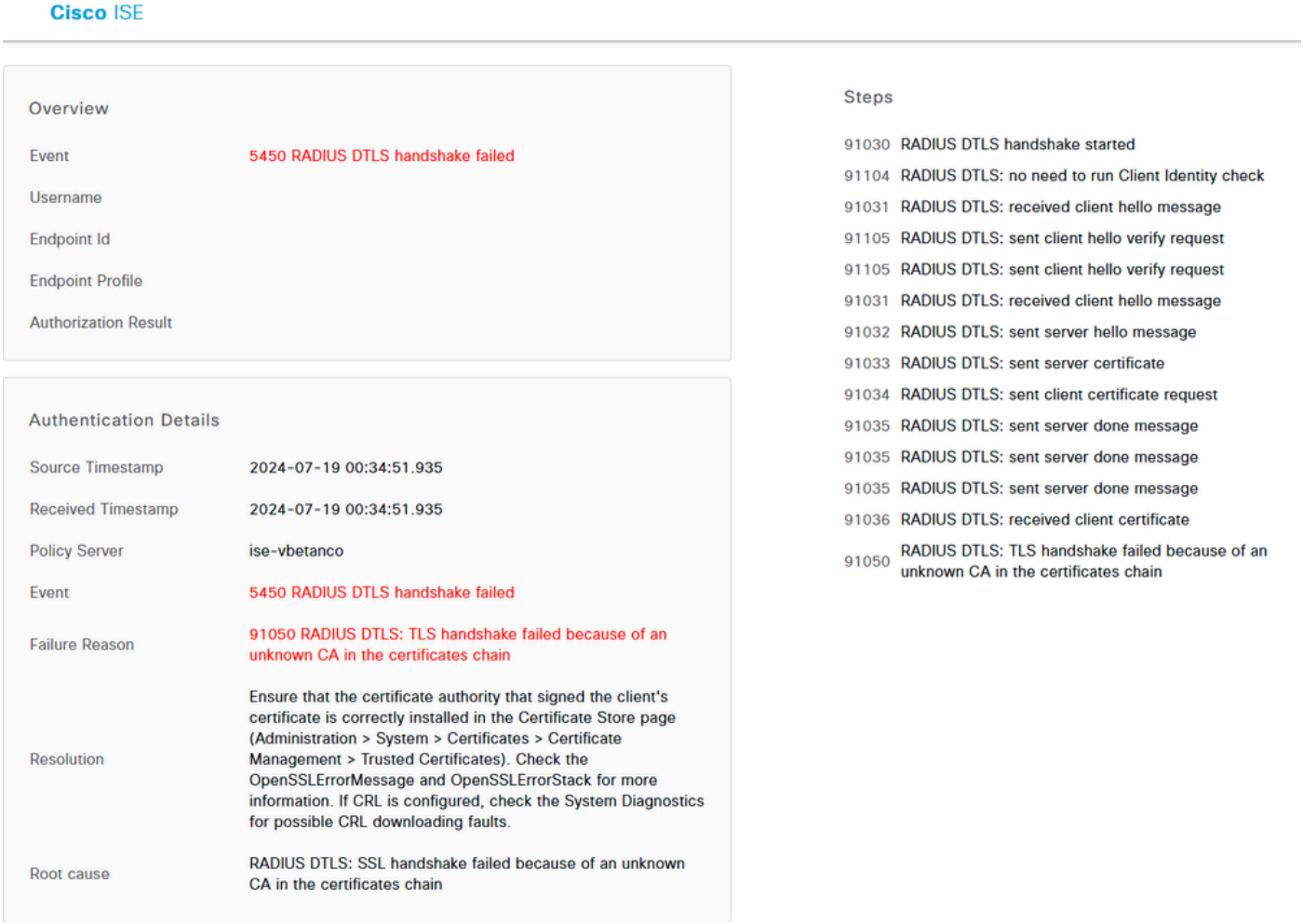
```
9800(config)#radius server
```

```
      9800(config)#dtls match-server-identity hostname
```

CA 인증서 체인을 컨트롤러에서 올바르게 가져오고 **dtls trustpoint server** configuration uses the Issuer CA trustpoint.

ISE에서 보고된 알 수 없는 CA

ISE가 WLC에서 제공하는 인증서를 검증할 수 없으면 DTLS 터널을 생성하지 못하고 인증이 실패합니다. 이는 RADIUS 라이브 로그에 오류로 표시됩니다. Operations(작업) >Radius>Live logs(라이브 로그)로 이동하여 확인합니다.



ISE 라이브 로그가 알 수 없는 CA로 인한 DTLS 핸드셰이크 실패를 보고합니다.

이 문제를 해결하려면 중간 인증서와 루트 인증서를 모두 확인하고 Administration(관리)>System(시스템)>Certificates(인증서)>Trusted certificates(신뢰할 수 있는 인증서)에서 Trust for client authentication and Syslog(클라이언트 인증 및 Syslog에 대한 신뢰) 확인란을 선택합니다.

폐기 검사 진행 중

인증서를 WLC로 가져올 때 새로 생성된 신뢰 지점에 폐기 검사가 활성화됩니다. 그러면 WLC에서 사용할 수 없거나 연결할 수 없는 인증서 해지 목록을 검색하려고 시도하고 인증서 확인에 실패합니다.
인증서에 대한 확인 경로의 각 신뢰 지점에 명령이 포함되어 있는지 확인합니다revocation-check none.

```
Jul 17 21:50:39.064: RADIUS_RADSEC_HASH_KEY_MATCH: hashkey1(0) matches hashkey2(0) TRUE
```

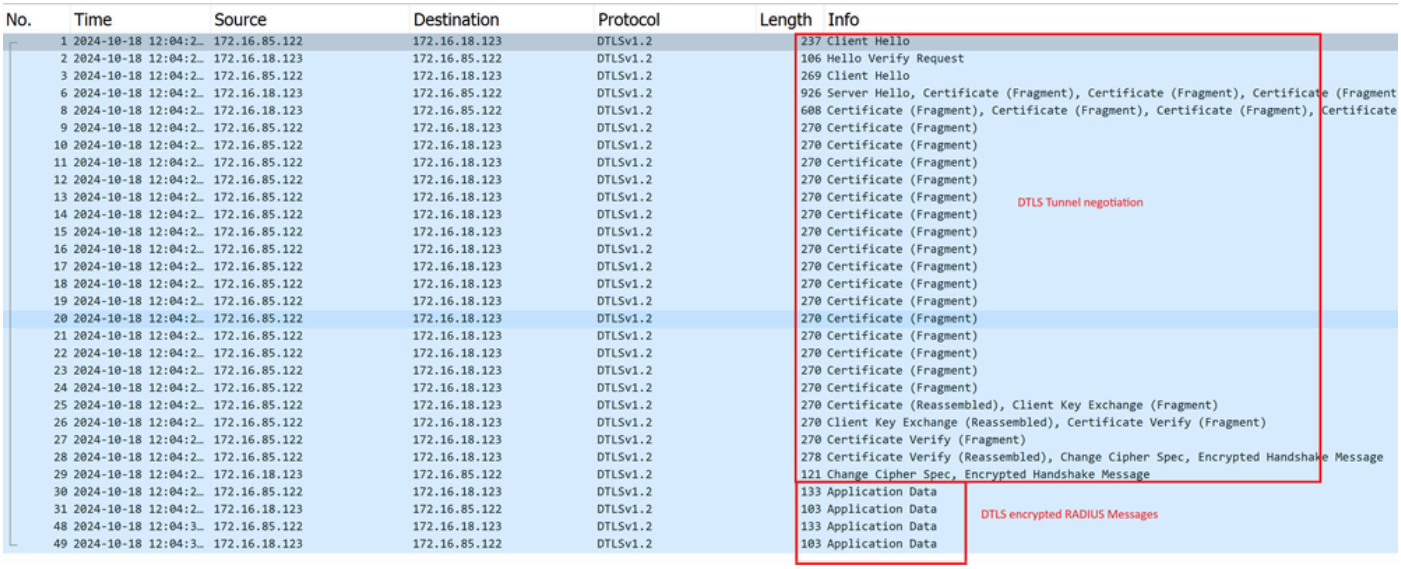
```
Jul 17 21:50:39.064: RADIUS_RADSEC_HASH_KEY_GET_CTX: radius radsec  sock_ctx(0x780FB0715978:0) get for
Jul 17 21:50:39.064: RADIUS_RADSEC_PROCESS_SOCK_EVENT: Handle socket event for TLS handshake(172.16.18.
Jul 17 21:50:39.064: RADIUS_RADSEC_STOP_TIMER: Stopped (172.16.18.123/2083)
Jul 17 21:50:39.068: %PKI-3-CRL_FETCH_FAIL: CRL fetch for trustpoint WLC1.pfx failed
                      Reason : Enrollment URL not configured. <------ WLC tries to perform revocation ch
Jul 17 21:50:39.070: RADIUS_RADSEC_HS_CONTINUE: TLS handshake failed!
Jul 17 21:50:39.070: RADIUS_RADSEC_UNQUEUE_WAIT_Q: Success Server(172.16.18.123)/Id(2)
Jul 17 21:50:39.070: RADIUS_RADSEC_FAILOVER_HANDLER:Failng-over to new server = 0x0
Jul 17 21:50:39.070: RADIUS_RADSEC_UNQUEUE_WAIT_Q: Empty Server(172.16.18.123)/Id(-1)
Jul 17 21:50:39.070: RADIUS_RADSEC_FAILOVER_HANDLER: no more data available
Jul 17 21:50:39.070: RADIUS_RADSEC_STOP_TIMER: Stopped (172.16.18.123/2083)
Jul 17 21:50:39.070: RADIUS_RADSEC_CONN_CLOSE: Cleaned up timers for Radius RADSEC ctx
Jul 17 21:50:39.070: RADIUS_RADSEC_GENERATE_HASHKEY: hash key(0) generated for sock(0)
Jul 17 21:50:39.070: RADIUS_RADSEC_GENERATE_HASHBUCKET: hash bucket(0) generated for sock(0)
Jul 17 21:50:39.070: RADIUS_RADSEC_HASH_KEY_DEL_CTX: remove [radius_radsec  ctx(0x780FB0715978)] succee
Jul 17 21:50:39.070: RADIUS_RADSEC_CONN_CLOSE: Hash table entry removed for RADSEC sock ctx
Jul 17 21:50:39.070: RADIUS_RADSEC_CONN_CLOSE: Success
Jul 17 21:50:39.070: RADIUS_RADSEC_SOCK_TLS_EVENT_HANDLE: Failed to complete TLS handshake
Jul 17 21:50:39.070: RADIUS_RADSEC_STOP_TIMER: Stopped (172.16.18.123/2083)
Jul 17 21:50:39.070: RADIUS_RADSEC_CONN_CLOSE: Cleaned up timers for Radius RADSEC ctx
Jul 17 21:50:39.070: RADIUS_RADSEC_GENERATE_HASHKEY: hash key(-1) generated for sock(-1)
Jul 17 21:50:39.070: RADIUS_RADSEC_GENERATE_HASHBUCKET: hash bucket(-1) generated for sock(-1)
Jul 17 21:50:39.070: RADIUS_RADSEC_HASH_KEY_DEL_CTX: remove [radius_radsec  ctx(0x780FB0715978)] succee
Jul 17 21:50:39.070: RADIUS_RADSEC_CONN_CLOSE: Hash table entry removed for RADSEC sock ctx
Jul 17 21:50:39.070: RADIUS_RADSEC_CONN_CLOSE: Success
Jul 17 21:50:39.070: RADIUS_RADSEC_SOCK_TLS_EVENT_HANDLE: Error
Jul 17 21:50:39.070: RADIUS_RADSEC_PROCESS_SOCK_EVENT: failed to hanlde radsec hs event
Jul 17 21:50:39.070: RADIUS_RADSEC_CLIENT_PROCESS: Got Socket Event
```

패킷 캡처에서 DTLS 터널 설정 문제 해결

9800 WLC는 EPC(Embedded Packet Capture) 기능을 제공하며, 이 기능을 사용하면 지정된 인터페이스에 대한 모든 전송 및 수신 트래픽을 캡처할 수 있습니다. ISE는 수신 및 발신 트래픽을 모니터링하기 위해 TCP 덤프라는 유사한 기능을 제공합니다. 동시에 사용할 경우 두 디바이스 모두의 관점에서 DTLS 세션 설정 트래픽을 분석할 수 있습니다.

ISE에서 TCP 덤프를 구성하는 자세한 단계는 Cisco Identity Services Engine 관리자 설명서를 참조하십시오. 또한 WLC에서 EPC 기능을 구성하는 방법에 대한 자세한 내용은 Catalyst 9800 Wireless LAN Controller 문제 해결을 참조하십시오.

성공적인 DTLS 터널 설정의 예입니다.



RADIUS DTLS 터널 협상 및 암호화된 메시지의 패킷 캡처

패킷 캡처는 DTLS 터널 설정이 수행되는 방식을 보여줍니다. 디바이스 간 손실된 트래픽 또는 DTLS 암호화 알림 패킷에서 협상에 문제가 있는 경우 패킷 캡처를 통해 문제를 식별할 수 있습니다 .

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.