ISEおよび9800 WLCでのRADIUS DTLSの設定

内容

はじめに

このドキュメントでは、ISEと9800 WLCの間でRADIUS DTLSを設定するために必要な証明書を作成する方法について説明します。

背景

RADIUS DTLSは、RADIUSメッセージがData Transport Layer Security(DTLS)トンネルを介して送信されるRADIUSプロトコルのセキュアな形式です。認証サーバとオーセンティケータの間にこのトンネルを作成するには、一連の証明書が必要です。この一連の証明書では、特定の拡張キー使用法(EKU)証明書拡張、特にWLC証明書でのクライアント認証、およびISE証明書のサーバ認証とクライアント認証の両方を設定する必要があります。

前提条件

要件

次の項目に関する知識があることが推奨されます。

- 9800 WLC、アクセスポイント(AP)の基本動作用の設定方法
- OpenSSL アプリケーションを使用する方法
- 公開キーインフラストラクチャ(PKI)とデジタル証明書

使用するコンポーネント

このドキュメントの情報は、次のソフトウェアとハードウェアのバージョンに基づいています。

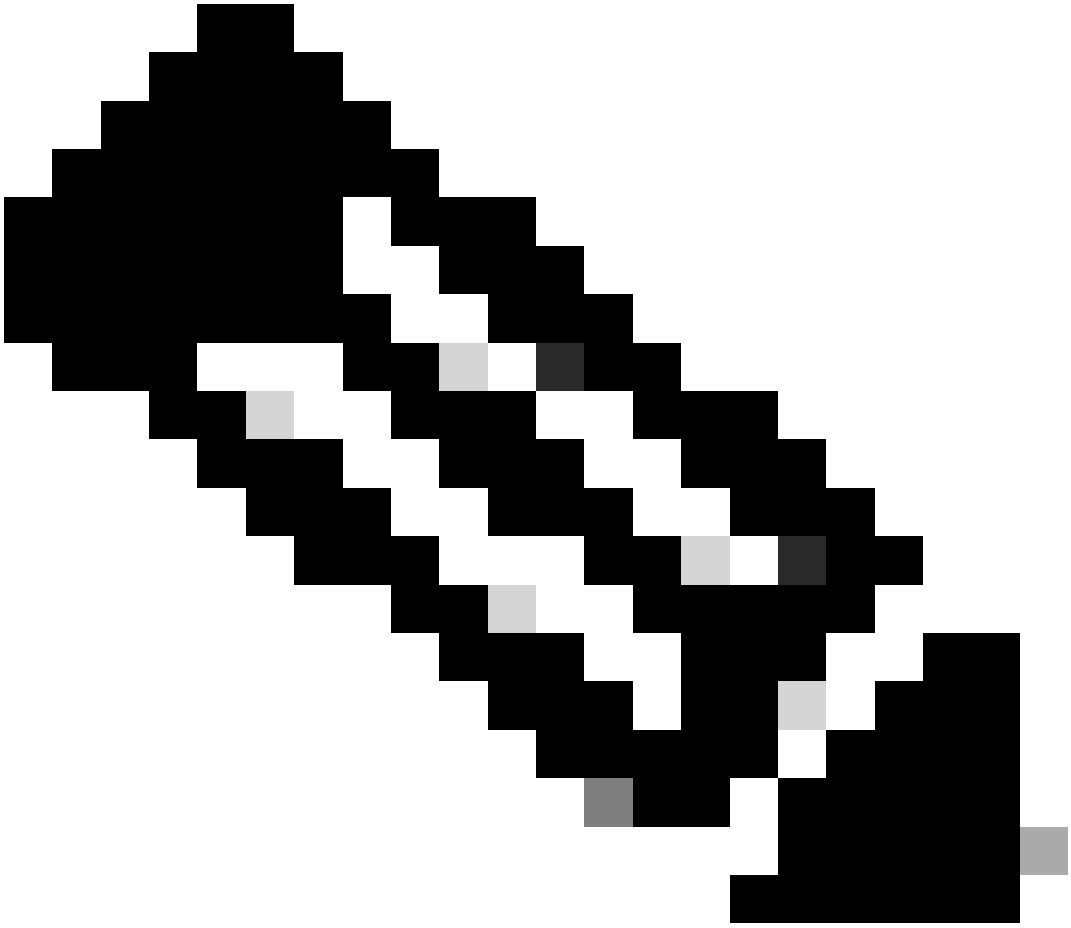
- OpenSSLアプリケーション（バージョン3.0.2）
- ISE（バージョン3.1.0.518）
- 9800 WLC（バージョン17.12.3）

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、クリアな（デフォルト）設定で作業を開始しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認してください。

設定

概要

目的は、ルートCAと中間CAを使用して2レベルの認証局を作成し、エンドポイント証明書に署名することです。証明書が署名されると、WLCとISEにインポートされます。最後に、これらの証明書を使用してRADIUS DTLS認証を実行するようにデバイスが設定されます。



注：このドキュメントでは、Linux固有のコマンドを使用してファイルを作成および配置します。OpenSSLを使用できる他のオペレーティングシステムでも同じ操作を実行できるように、コマンドについて説明します。

オプション：WLCおよびISE RADIUS DTLSデバイス証明書の作成

RADIUS DTLSプロトコルは、DTLSトンネルを作成するためにISEとWLCの間で証明書を交換する必要があります。有効な証明書がない場合は、ローカルCAを作成して証明書を生成できます。「CIsco IOS® XE互換証明書を生成するためのOpenSSLでのマルチレベル認証局の設定」を参照し、このドキュメントで説明されている手順を最初から手順の最後まで実行します 中間CA証明書を作成します。

openssl.cnfファイルへの設定セクションの追加

openssl.cnfコンフィギュレーションファイルを開き、ファイルの下部に、有効な証明書署名要求(CSR)の生成に使用したWLCとISEのセクションをコピーアンドペーストします。

ISE_device_req_extセクションとWLC_device_req_extセクションの両方とも、CSRに含めるSANのリストを指しています。

```
#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
```

セキュリティ対策として、CAは、CSRに署名するためにCSRに存在するすべてのSANを上書きするため、許可されていないデバイスは、使用が許可されていない名前の有効な証明書を受け取ることができません。署名付き証明書にSANを追加し直すには、subjectAltNameパラメータを使用して、CSRの生成に使用したものと同じリストSANを指し示します。

ISEでは、証明書に存在するserverAuthとclientAuthの両方のEKUが必要ですが、WLCではclientAuthのみが必要です。これらはextendedKeyUsage パラメータを使用して署名付き証明書に追加されます。

証明書の署名に使用するセクションをopenssl.cnfファイルの下部にコピーアンドペーストします。

```
#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デバイス証明書の作成

IntermCA.db.certsという名前の中間CA証明書フォルダ内にOpenSSLがインストールされているマシン上にWLC証明書を保存する新しいフォルダを作成します。この新しいフォルダはWLCです。

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

openssl.cnfファイルの[WLC_alt_names]セクションでDNSパラメータを変更します。指定した名前の例を目的の値に変更します。次の値がWLC証明書のSANsフィールドに入力されます。

```
[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
```

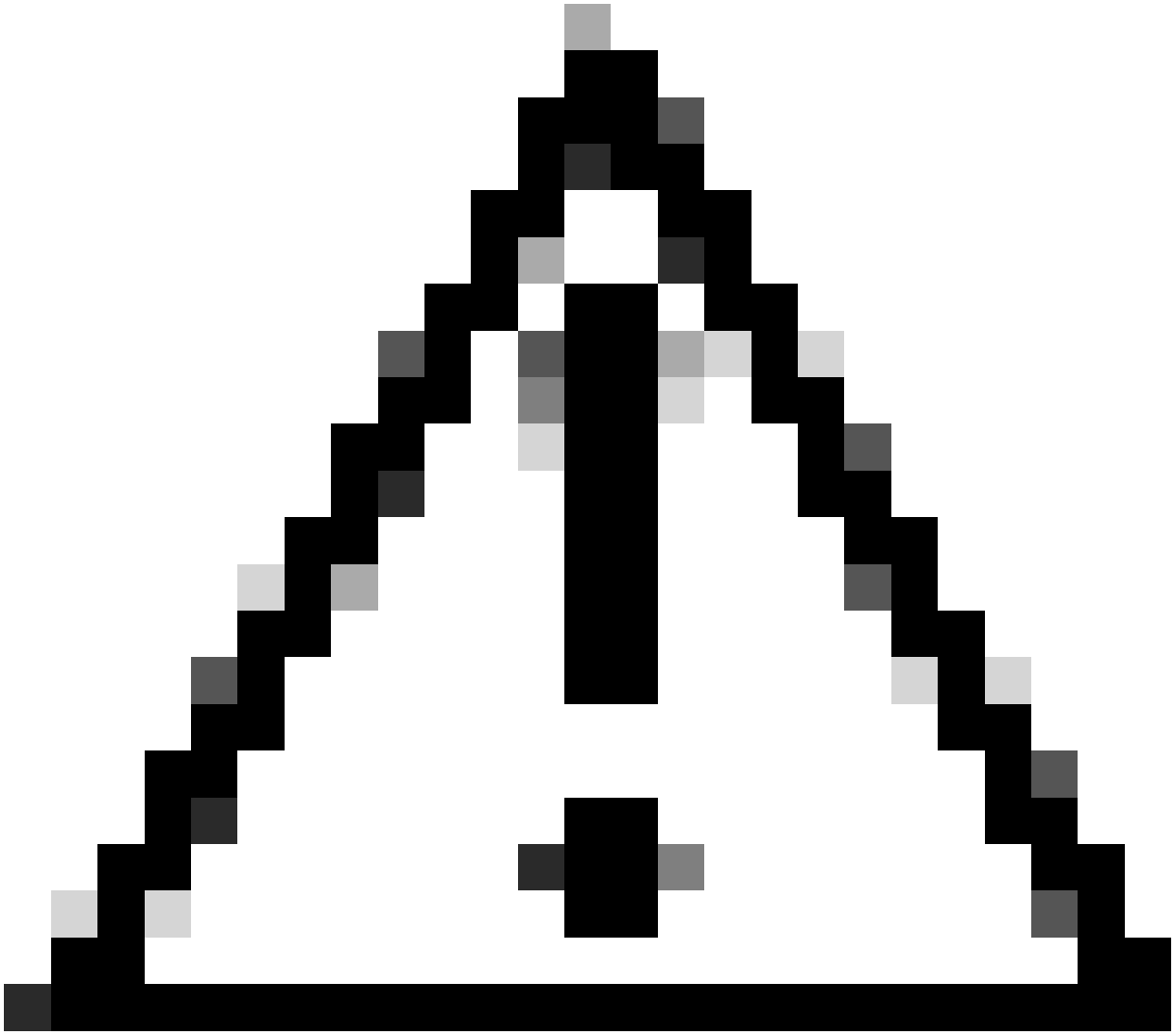
WLC秘密キーとWLC CSRを、セクションWLC_device_req_extからの情報を使用してSAN用に作成します。

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

OpenSSLは、識別名(DN)の詳細を入力するためのインタラクティブなプロンプトを開きます。

```
.........+..+.......+..........+...+.....+.+...........+..+++++++++++++
++++++
-----
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証明書識別名インタラクティブプロンプト



注意：インタラクティブなプロンプトで指定する共通名(CN)は、openssl.cnfファイルの[WLC_alt_names]セクションの名前のいずれかと同じである必要があります。

IntermCAという名前のCAを使用して、[WLC_cert]で定義された拡張子を持つWLC.csrという名前のWLC CSRに署名し、署名済み証明書を./IntermCA/IntermCA.db.certs/WLC内に保存します。WLCデバイス証明書はWLC.crtと呼ばれます。

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

9800 WLCで証明書をインポートするには、証明書がpfx形式である必要があります。WLC証明書に署名したCAのチェーンを含む新しいファイルを作成します。これはcertfileと呼ばれます。

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

.pfxファイルを作成するには、WLCのバージョンに応じて、次のコマンドのいずれかを実行します。

17.12.1より前のバージョンの場合：

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

バージョン17.12.1以降：

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

ISEデバイス証明書の作成

IntermCA.db.certsという名前の中間CA証明書フォルダ内にOpenSSLがインストールされているマシン上にISE証明書を保存するための新しいフォルダを作成します。この新しいフォルダはISEです。

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

openssl.cnfファイルの[ISE_alt_names]セクションでDNSパラメータを変更します。指定した名前の例を目的の値に変更します。次の値がWLC証明書のSANsフィールドに入力されます。

```
[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
```

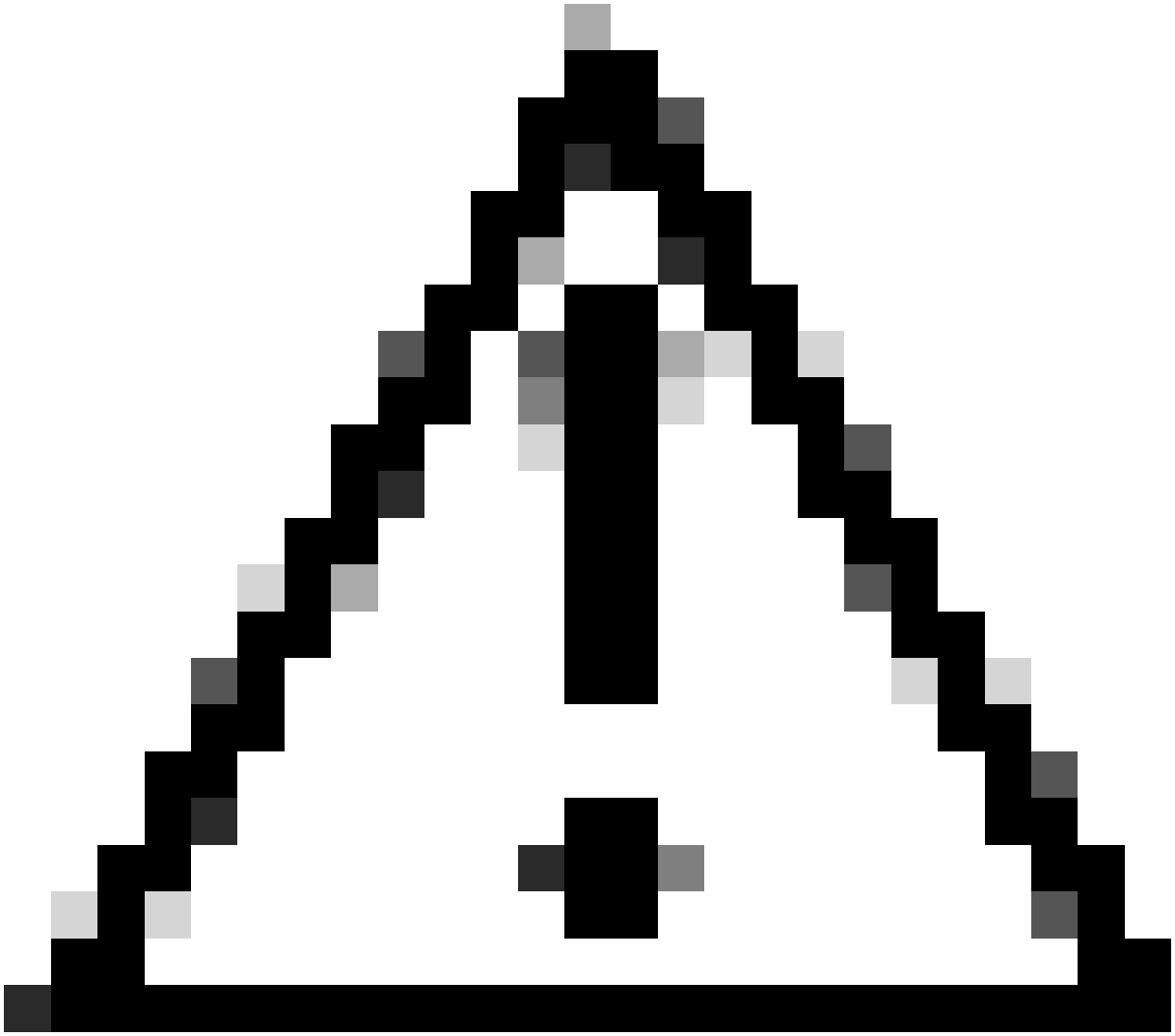
ISE秘密キーとISE CSRを、ISE_device_req_ext for SANsセクションの情報を使用して作成します。

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

OpenSSLは、識別名(DN)の詳細を入力するためのインタラクティブなプロンプトを開きます。

```
-----
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証明書識別名インタラクティブプロンプト



> 注意：インタラクティブなプロンプトで指定するCNは、openssl.cnfファイルの[ISE_alt_names]セクションの名前とまったく同じである必要があります。

IntermCAという名前のCAを使用して、[ISE_cert]で定義された拡張子を持つISE.csrという名前のISE CSRに署名し、署名済み証明書を./IntermCA/IntermCA.db.certs/WLC内に保存します。ISEデバイス証明書はISE.crtと呼ばれます。

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

デバイスへの証明書のインポート

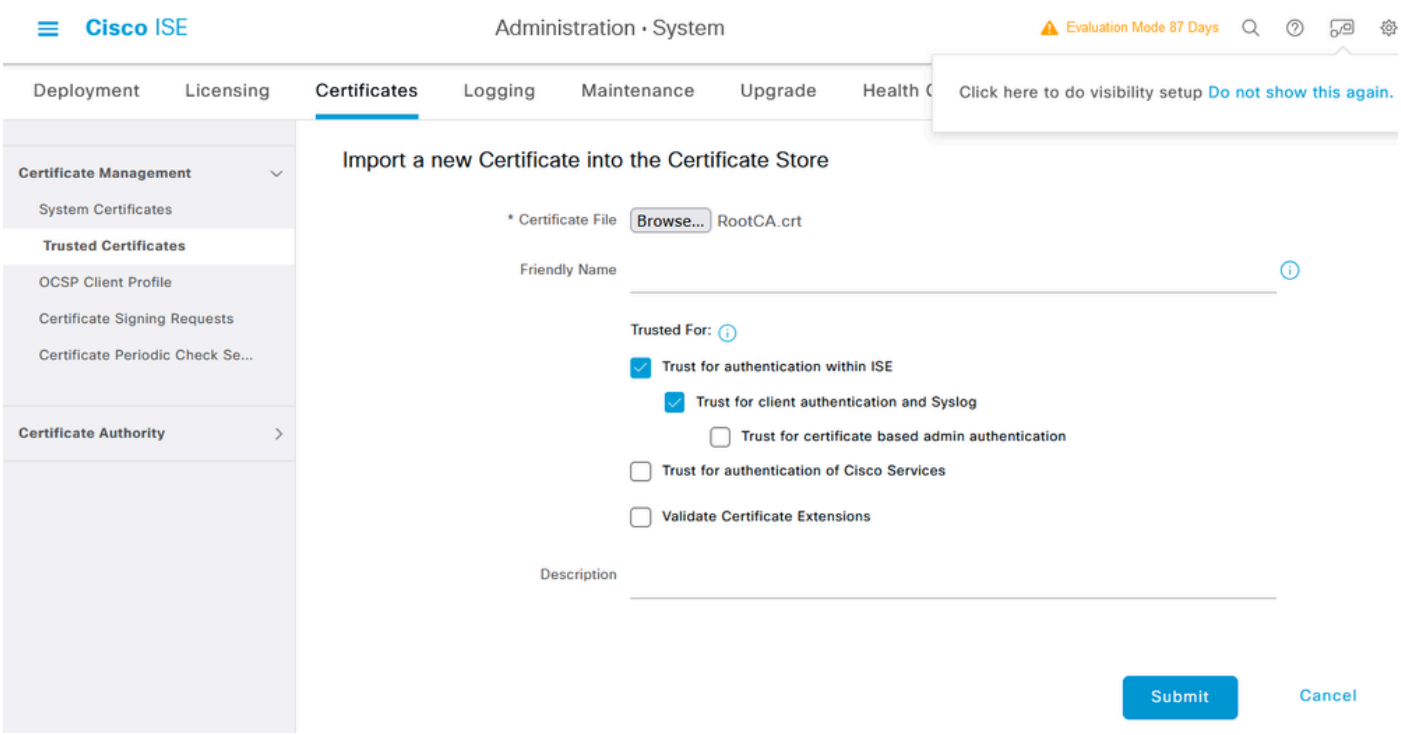
ISEへの証明書のインポート

1. ISE証明書チェーンから信頼できる証明書ストアにルートCA証明書をインポートします。

2. Administration > System > Certificates > Trusted certificatesの順に移動します。

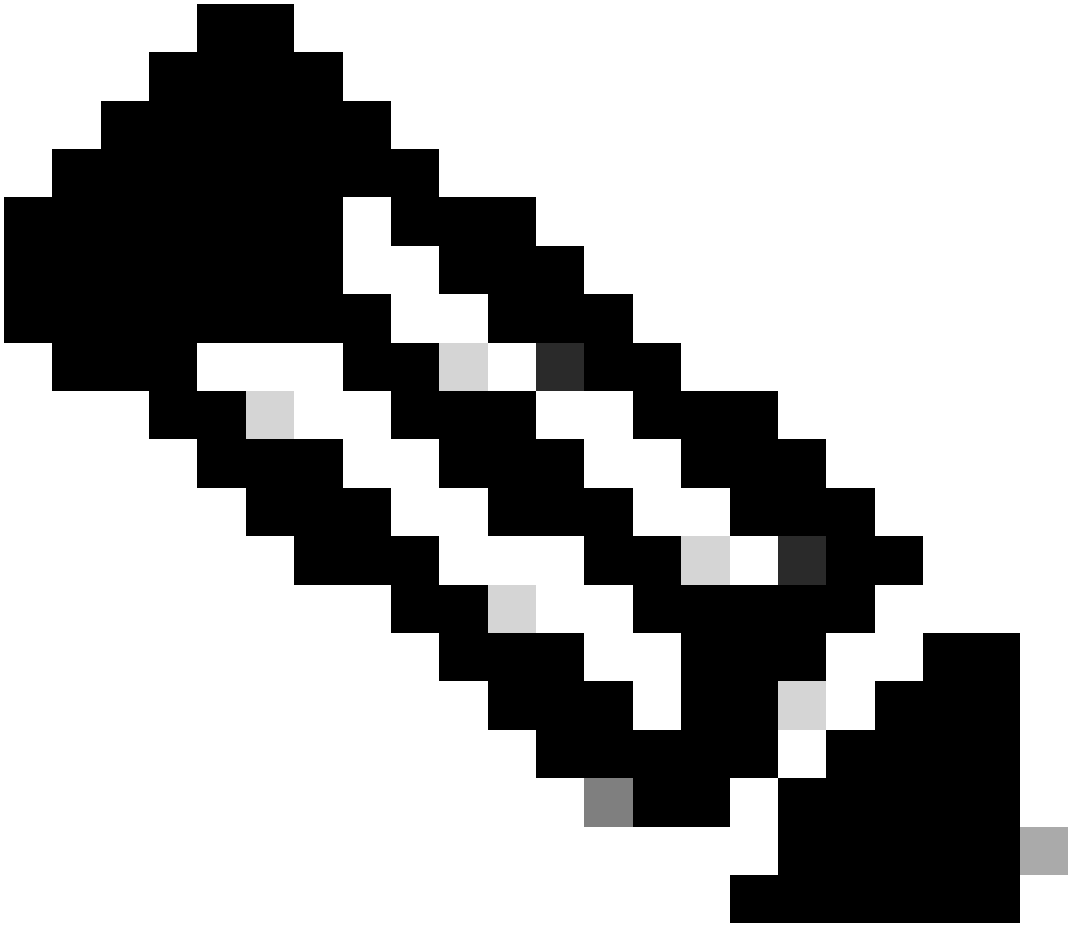
3. 「参照」をクリックし、Root.crtファイルを選択します。

4. Trust for authentication within ISEおよびTrust for client authentication and Syslogチェックボックスにチェックマークを入れて、Submitをクリックします。

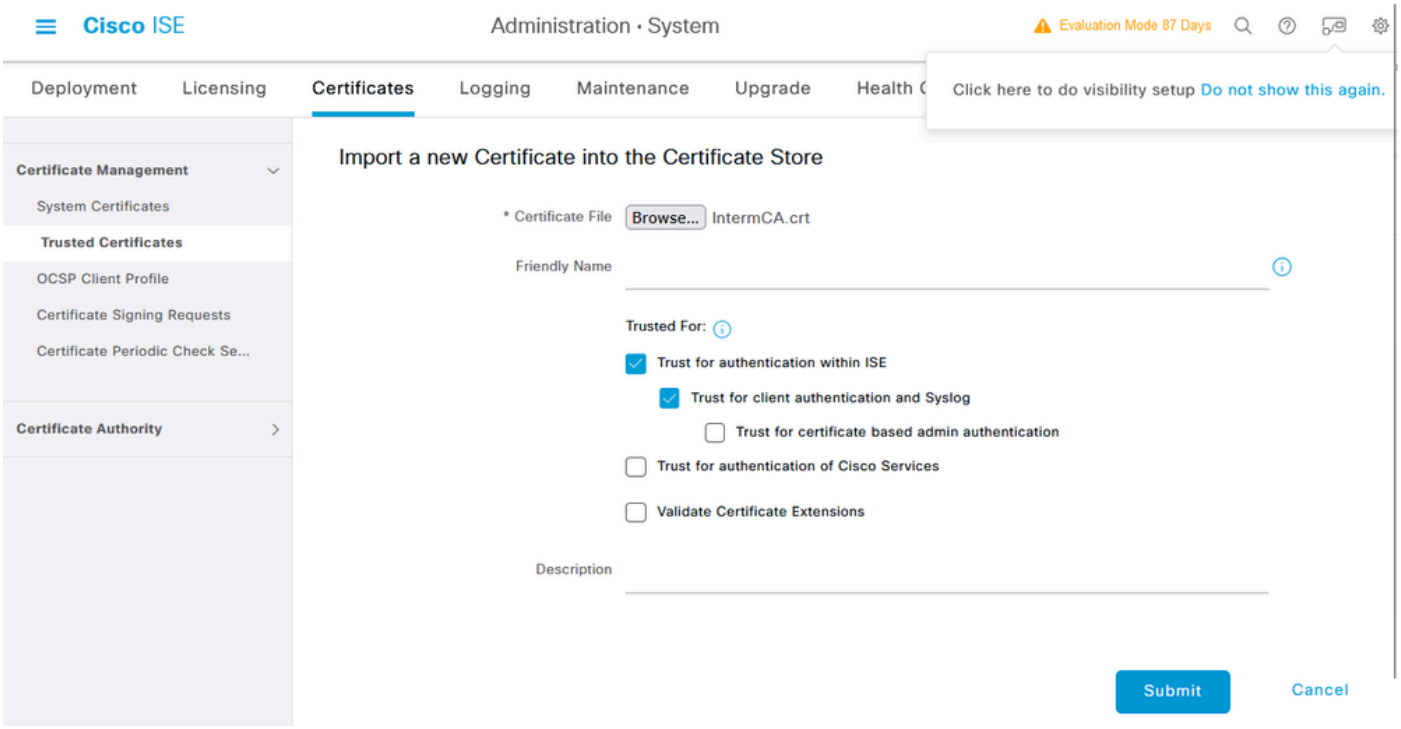


ISEルートCA証明書インポートダイアログ

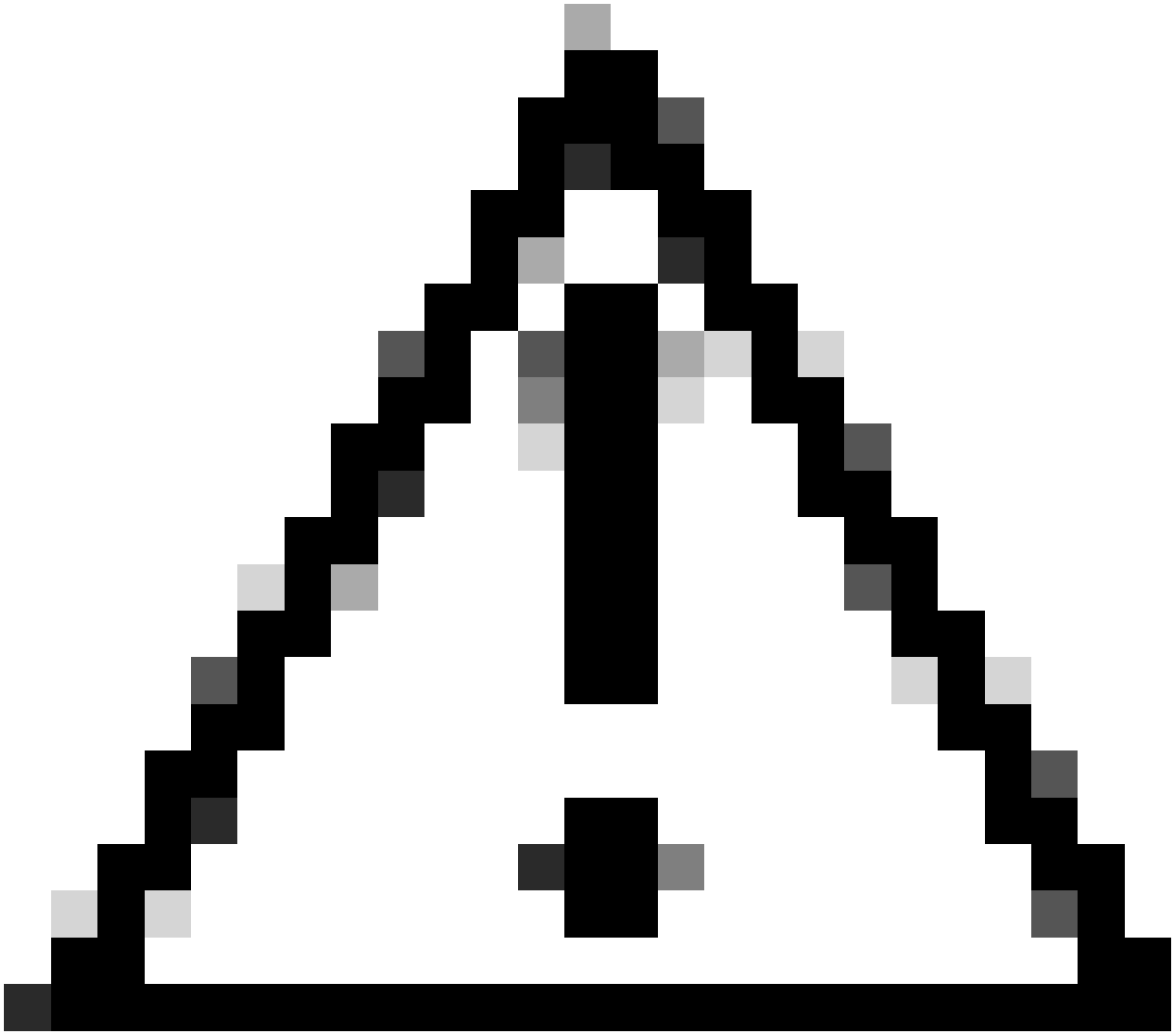
中間証明書が存在する場合は、中間証明書についても同じ手順を実行します。



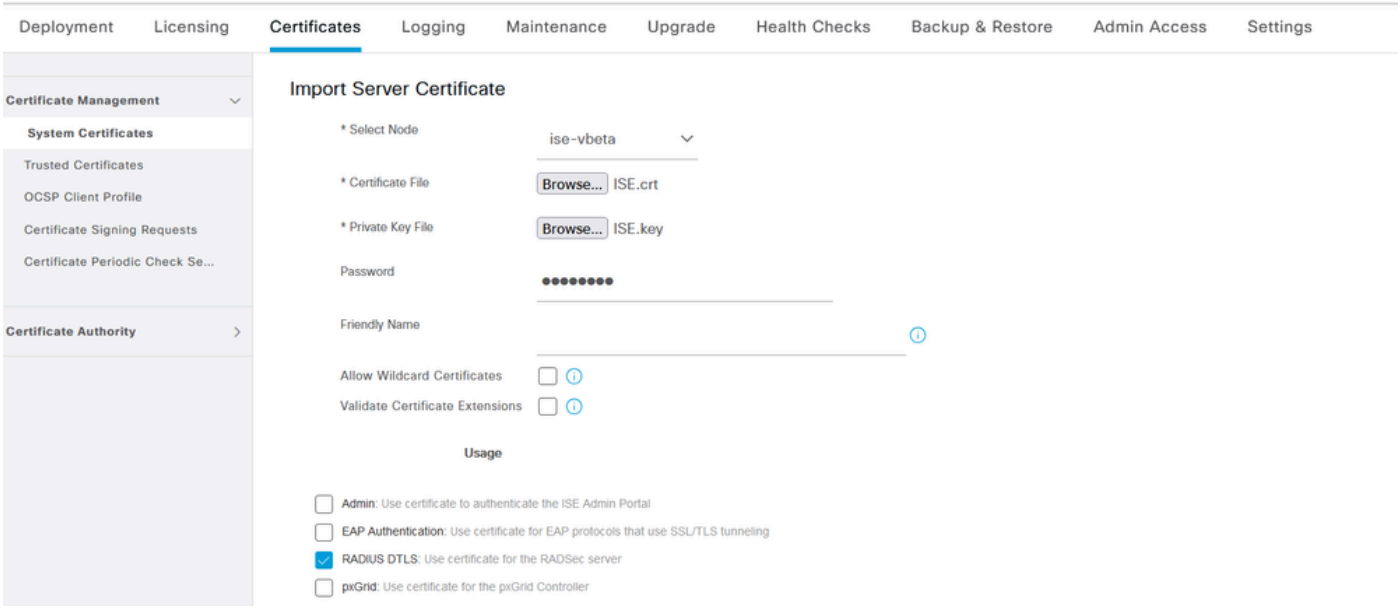
注:ISE証明書検証チェーンの一部であるCA証明書に対して、この手順を繰り返します。必ずルートCA証明書から開始し、チェーンの最も低い中間CA証明書で終了します。



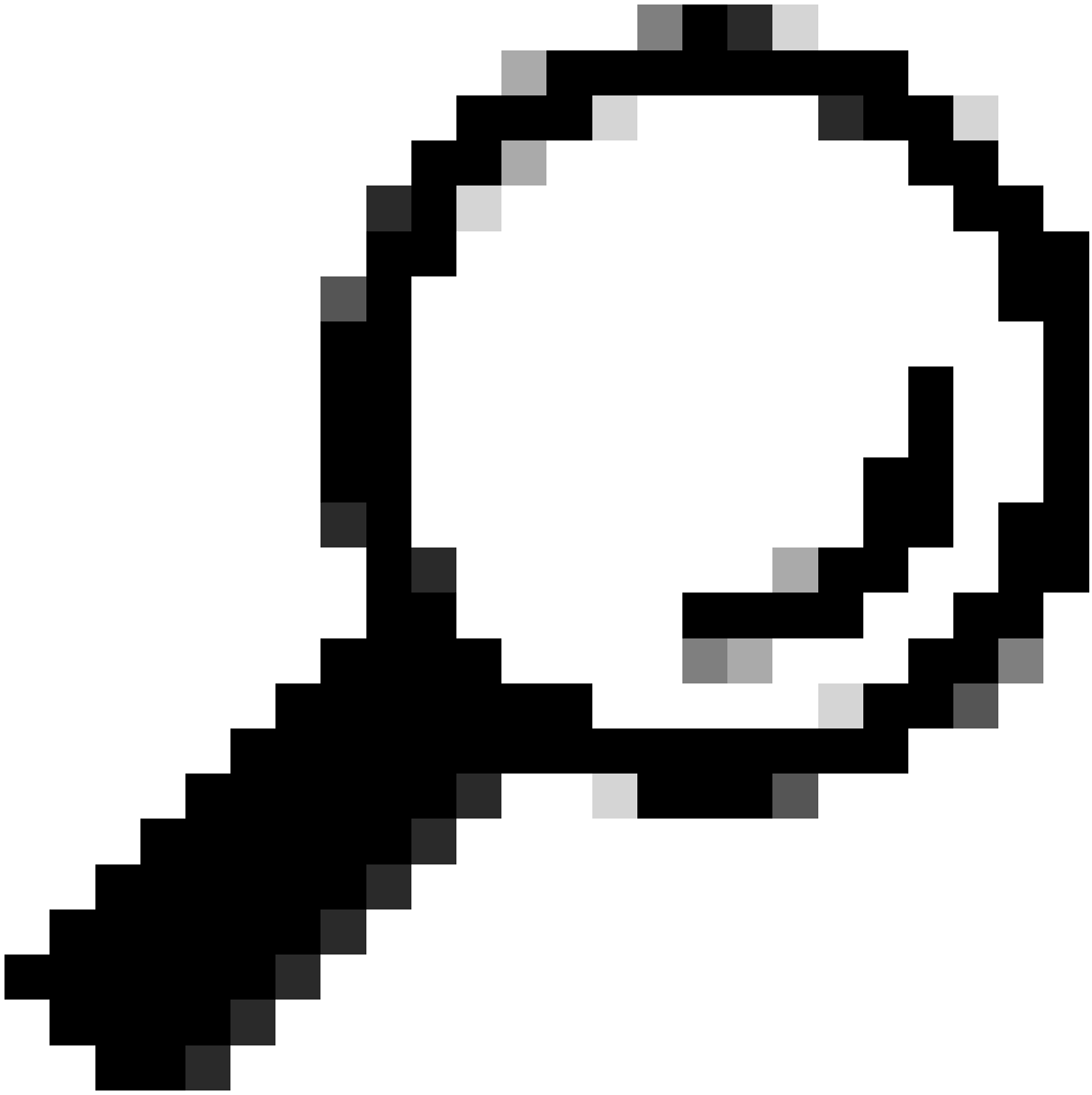
ISE中間CA証明書インポートダイアログ



注意:ISE証明書とWLC証明書が異なるCAによって発行されている場合、WLC証明書チェーンに属するすべてのCA証明書もインポートする必要があります。ISEは、これらのCA証明書をインポートするまで、DTLS証明書交換でWLC証明書を受け入れません。



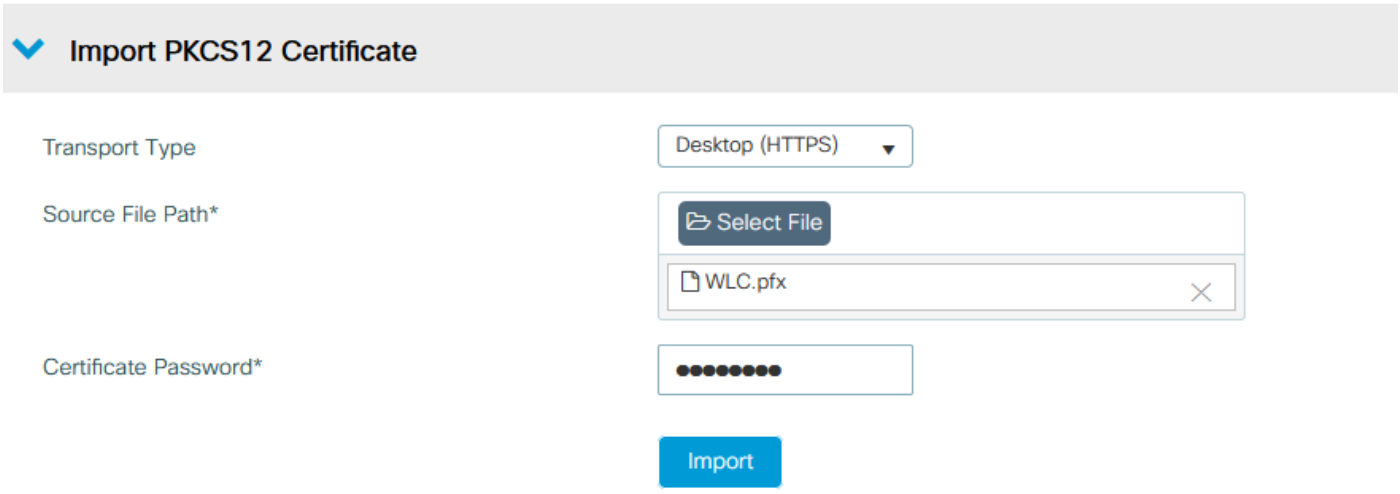
ISEデバイス証明書インポートメニュー



ヒント：このステップでインポートする必要があるのは、ISEデバイス証明書だけです。この証明書は、DTLSトンネルを確立するためにISEが交換する証明書です。WLC証明書は以前にインポートされたCA証明書を使用して検証されるため、WLCデバイス証明書と秘密キーをインポートする必要はありません。

WLCへの証明書のインポート

1. WLCでConfiguration > Security > PKI Managementの順に選択し、Add Certificateタブに移動します。
2. Import PKCS12 Certificate ドロップダウンをクリックし、トランスポートタイプをDesktop (HTTPS)に設定します。
3. Select Fileボタンをクリックし、前に準備した.pfxファイルを選択します。
4. インポートパスワードを入力して、最後にImportをクリックします。

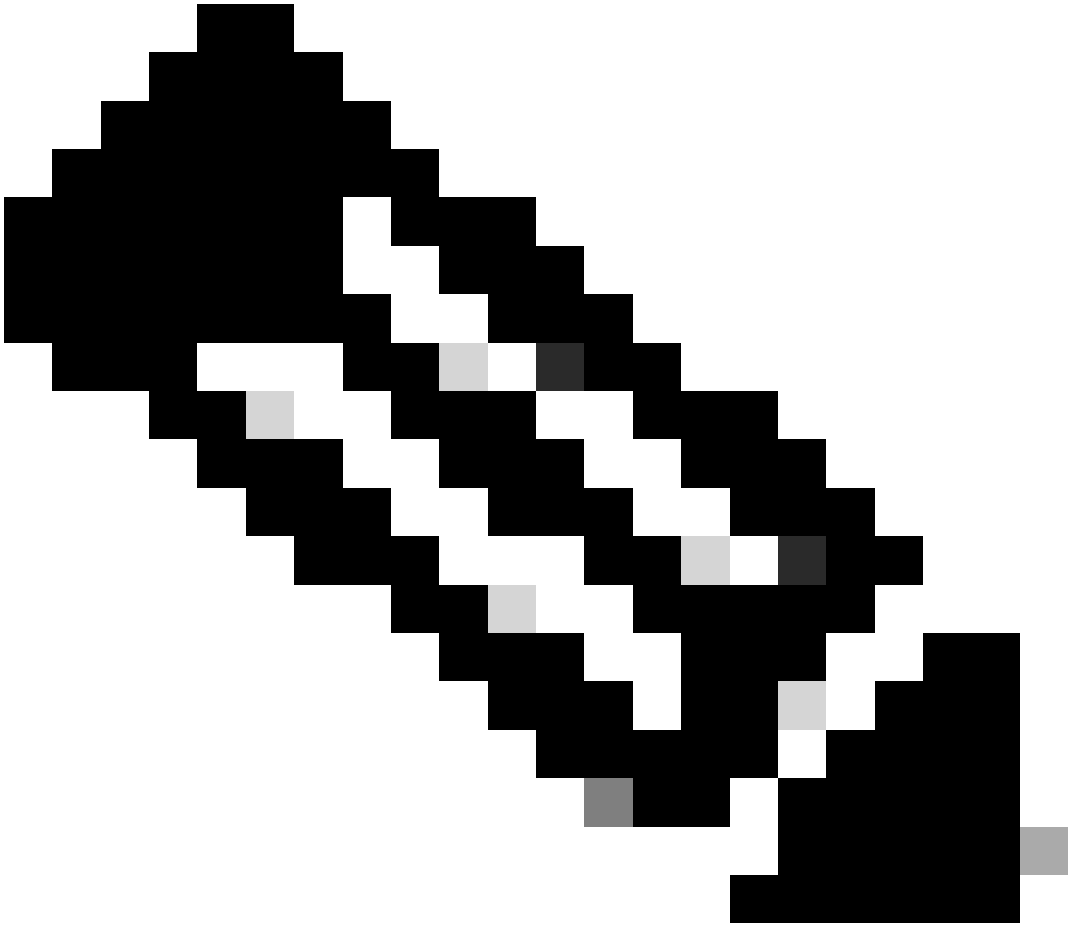


WLC証明書インポートダイアログ

インポートプロセスの詳細については、『Catalyst 9800 WLCでのCSR証明書の生成とダウンロード』を参照してください。

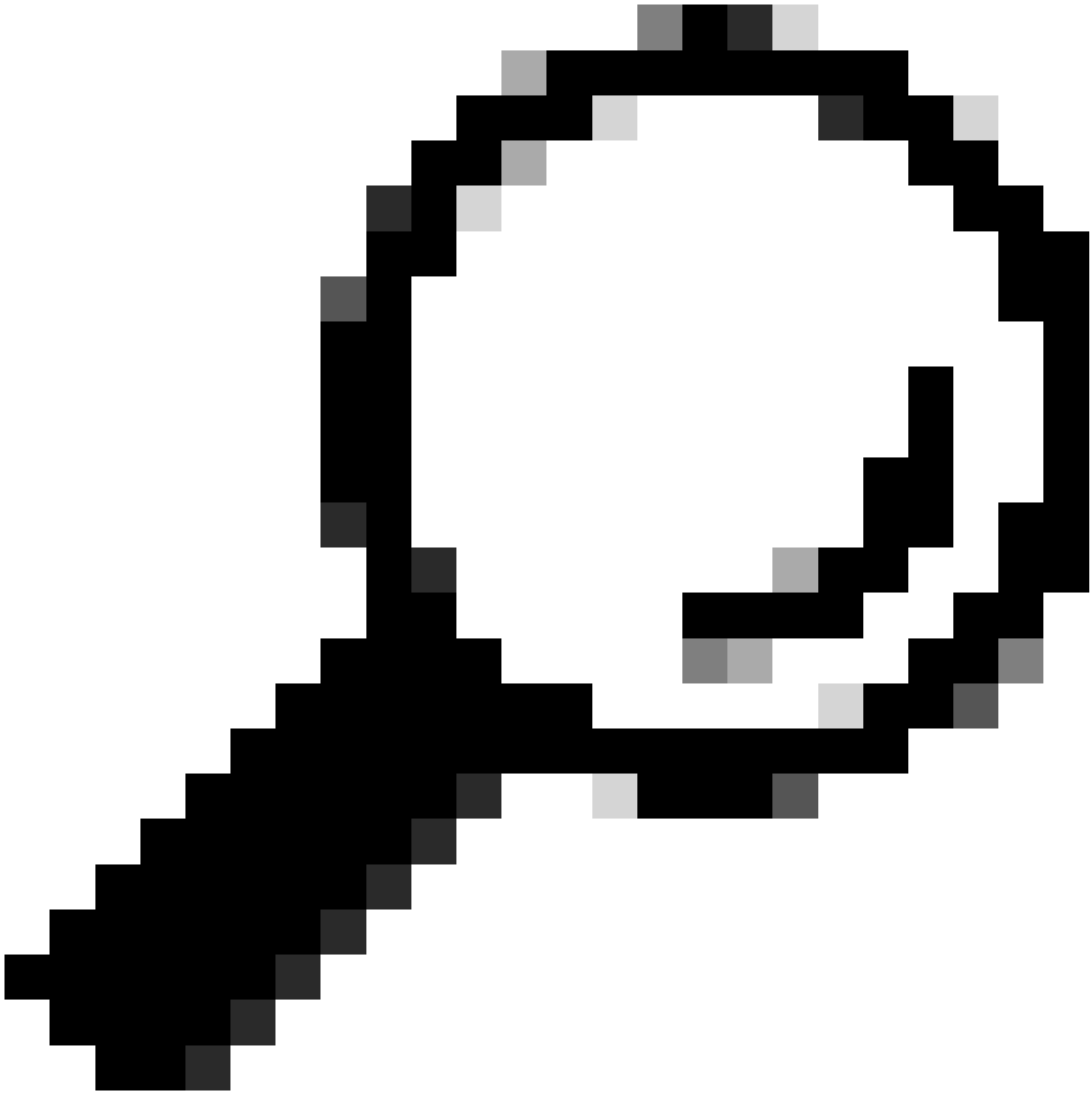
WLCに、ネットワークを介して確認できる証明書失効リスト(CRL)がない場合、自動的に作成された各トラストポイント内の失効チェックを無効にします。

```
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
```



注：『Configure Multi-Level CA on OpenSSL to Generate Cisco IOS XE Certificates』ドキュメントを参照してマルチレベルCAをOpenSSLで作成した場合、CRLサーバが作成されないため、失効チェックを無効にする必要があります。

自動インポートでは、WLC証明書とそのCA証明書を格納するために必要なトラストポイントが作成されます。



ヒント:WLC証明書がISE証明書と同じCAによって発行された場合は、WLC証明書インポートで自動的に作成されたものと同じトラストポイントを使用できます。ISE証明書を個別にインポートする必要はありません。

WLC証明書がISE証明書とは異なるCAによって発行される場合、ISEデバイス証明書を信頼するために、ISE CA証明書をWLCにインポートする必要もあります。
ルートCAの新しいトラストポイントを作成し、ISEルートCAをインポートします。

```
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チェーン上の次の中間CA証明書、つまりルートCAによって発行されたCA証明書をインポートします。

```
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-------
```

チェーン上の追加の各CAには、個別のトラストポイントが必要です。チェーン内の各トラストポイントは、コマンドchain-validation continue <Issuer trustpoint name>を使用して、インポートする証明書の発行者証明書を含むトラストポイントを参照する必要があります。

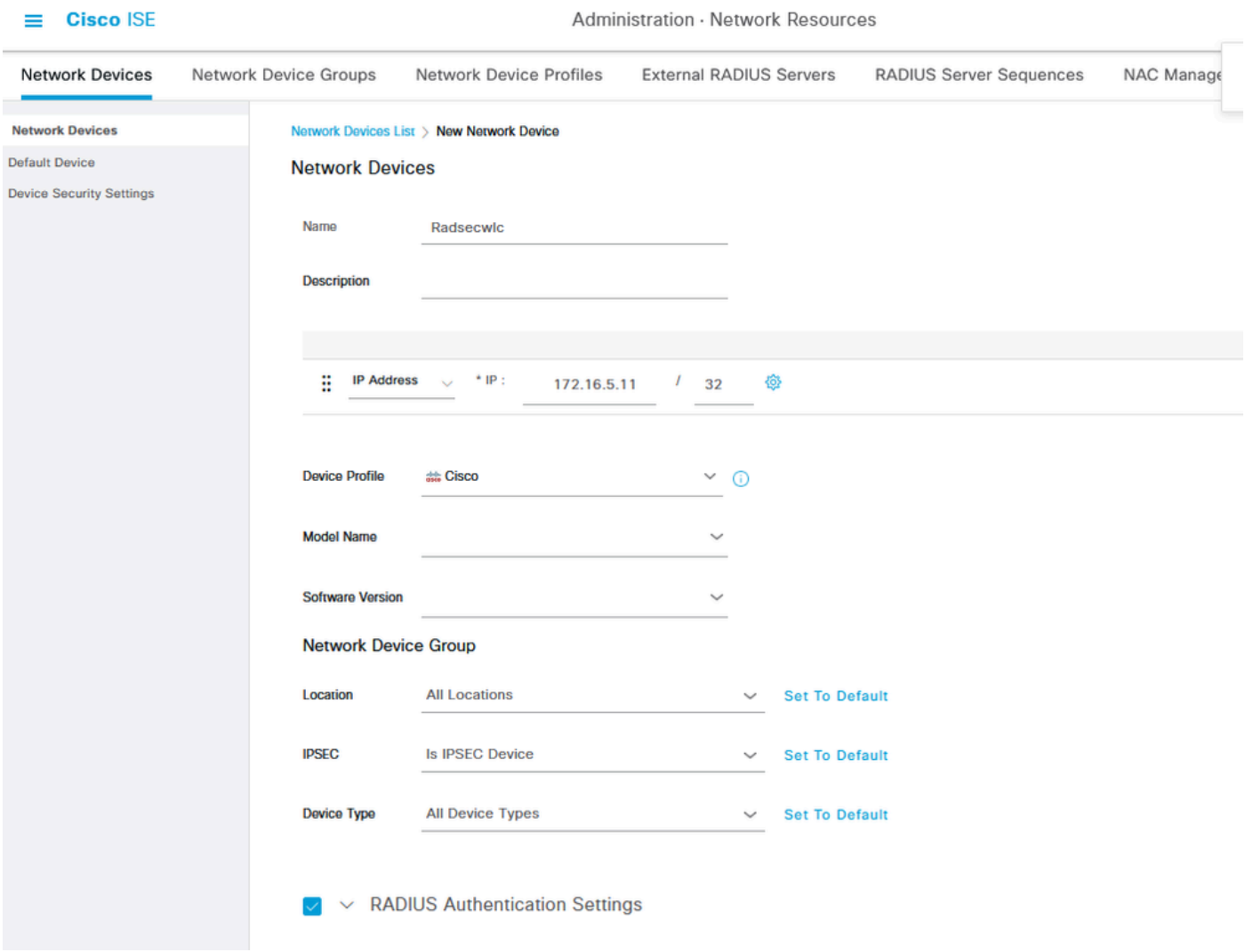
CAチェーンに含まれている数だけCA証明書をインポートします。ISEデバイス証明書の発行者CAのインポートが完了したら、このトラストポイントの名前を書き留めます。

RADIUS DTLSを機能させるためにWLC上でISEデバイス証明書をインポートする必要はありません。

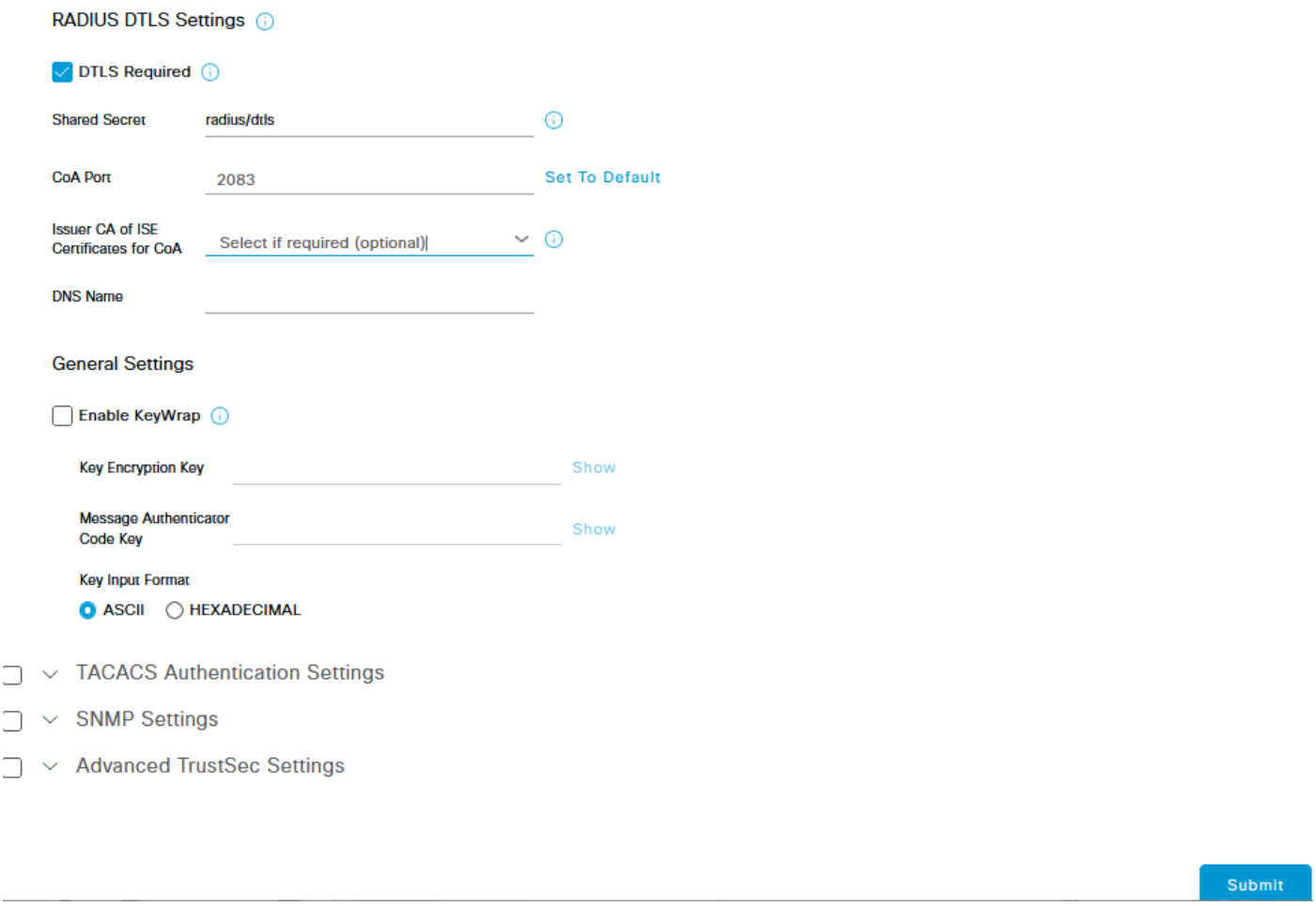
RADIUS DTLSの設定

ISE 設定

WLCをネットワークデバイスとしてISEに追加するには、Administration > Network Resources > Network devices > Addの順に移動します。
デバイスの名前と、RADIUSトラフィックを送信するWLCインターフェイスのIPを入力します。通常はワイヤレス管理インターフェイスIP。下にスクロールして、RADIUS Authentication SettingsとDTLS Requiredにチェックマークを入れて、Submitをクリックします。



新しいネットワークデバイスの設定



ISE上のネットワークデバイスのRADIUS DTLS設定

WLC の設定

ISE IPアドレスおよびRadius DTLSのデフォルトポートとともに新しいRADIUSサーバを定義します。この設定は、CLIでのみ使用できます。

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

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

Radius DTLSでは共有秘密鍵radius/dtlsを使用する必要があり、9800 WLCでは、このキー以外の設定済みキーは無視されます。

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

dtls trustpoint client
コマンドを使用して、DTLSトンネルと交換するWLCデバイス証明書が含まれるトラストポイントを設定します。
dtls trustpoint server
コマンドを使用して、ISEデバイス証明書用の発行者CAを含むトラストポイントを設定します。

クライアントとサーバのトラストポイント名は、WLCとISEの証明書が同じCAによって発行される場合にのみ同じになります。

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

ISE証明書に存在するサブジェクト代替名(SAN)の1つを確認するようにWLCを設定します。この設定は、証明書のSANsフィールドに存在するSANの1つと正確に一致している必要があります。

9800 WLCでは、SANフィールドに対する正規表現ベースの照合は実行されません。つまり、たとえば dtls match-server-identity hostname *.example.com、ワイルドカード証明書のSANフィールドに *.example.comがある場合のコマンドは正しいにも関わらず、同じコマンドのSANフィールドに www.example.comがある場合は正しくないことを意味します。

WLCでは、この名前をネームサーバに対してチェックしません。

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

新しいRadius DTLSを認証に使用する新しいサーバグループを作成します。

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

これ以降は、WLC上の他のサーバグループと同様に、このサーバグループを使用できます。このサーバをワイヤレスクライアント認証に使用するには、『Catalyst 9800ワイヤレスコントローラシリーズでの802.1X認証の設定』を参照してください。

確認

証明書情報の確認

作成した証明書の証明書情報を確認するには、Linuxターミナルで次のコマンドを実行します。

```
openssl x509 -in

      -text -noout
```

完全な証明書情報が表示されます。これは、特定の証明書の発行者CAを特定する場合や、証明書に必要なEKUおよびSANが含まれているかどうかを確認する場合に役立ちます。

```
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で表示されるCisco IOS XEデバイスの証明書情報

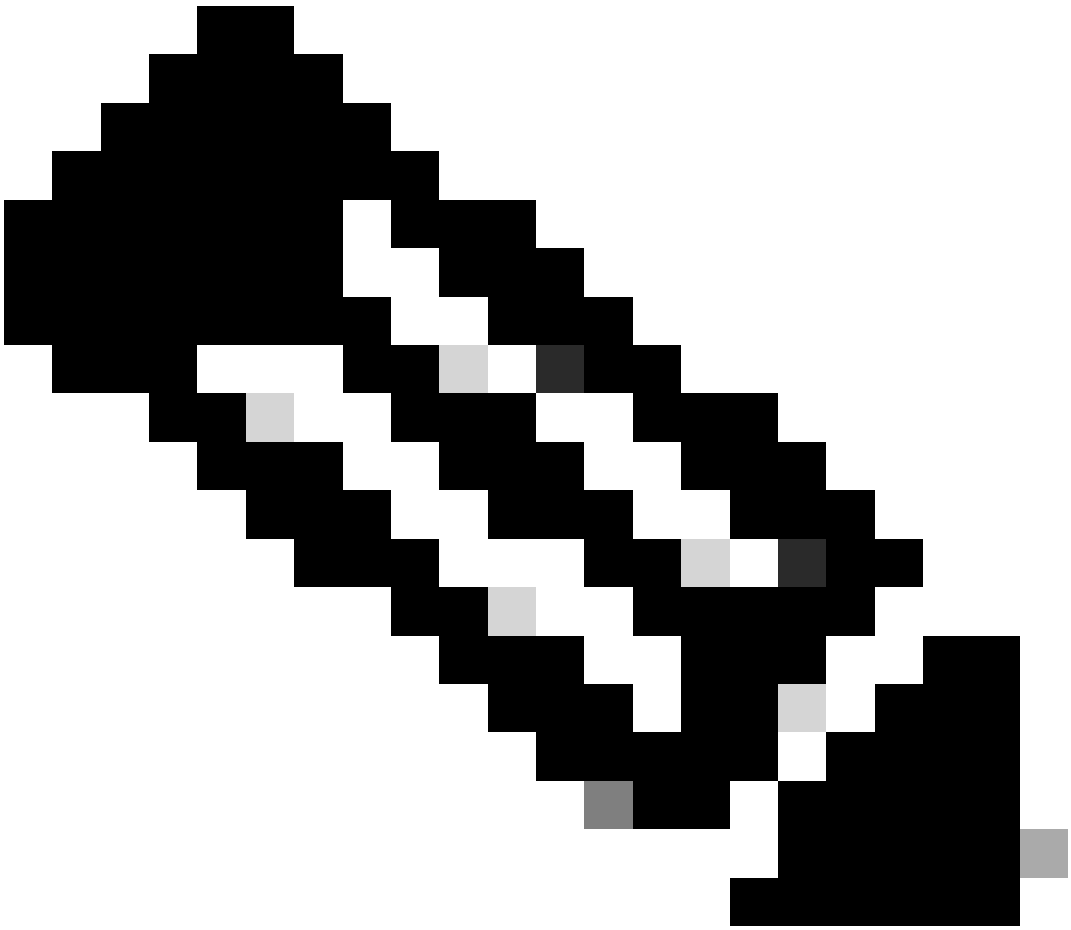
テスト認証の実行

WLCから、コマンドを使用してRadius DTLS機能をテストできます test aaa group

new-code

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

```
USER ATTRIBUTES

username              0   "testuser"
```



> 注：testコマンドの「access reject」出力は、WLCがAccess-Reject RADIUSメッセージを受信したことを意味しています。この場合、RADIUS DTLSは機能しています。ただし、これはDTLSトンネルの確立に失敗したことを示している場合もあります。testコマンドでは両方のシナリオが区別されません。トラブルシューティングのセクションを参照して、問題があるかどうかを確認してください。

トラブルシュート

認証の失敗の原因を確認するには、テスト認証を実行する前にこれらのコマンドを有効にします。

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

次に、デバッグを有効にした状態での認証の成功の出力を示します。

```
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
```

WLCによって報告される不明なCA

WLCがISEから提供された証明書を検証できない場合、WLCはDTLSトンネルの作成に失敗し、認証は失敗します。
この場合に表示されるデバッグメッセージの例を次に示します。

```
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:
```

これを修正するには、WLCに設定されたIDが、ISE証明書に含まれるSANの1つと正確に一致することを確認します。

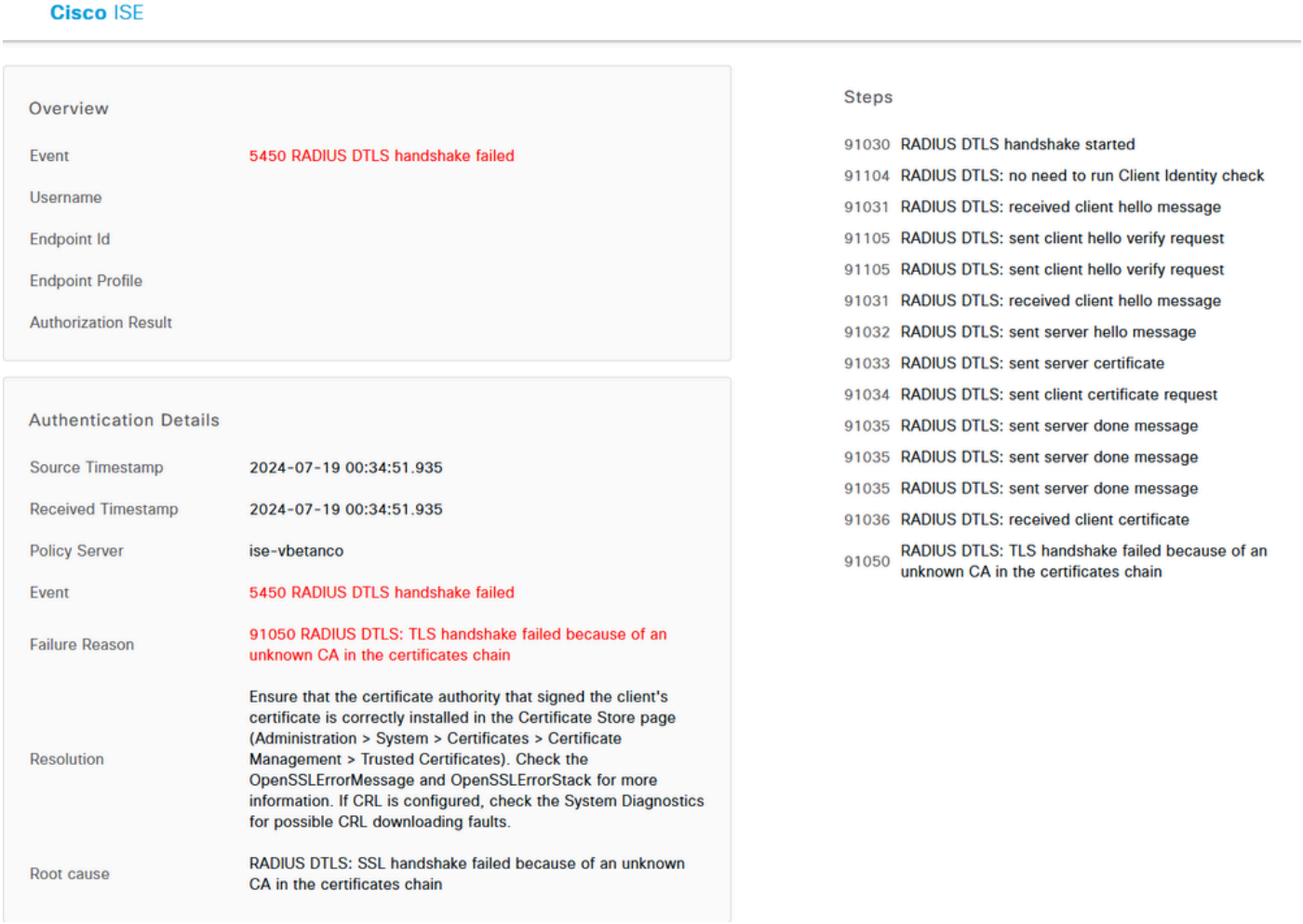
```
9800(config)#radius server
```

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

CA証明書チェーンがコントローラに正しくインポートされ、 **dtls trustpoint server** configuration uses the Issuer CA trustpoint.

ISEによって報告された不明なCA

ISEがWLCによって提供された証明書を検証できない場合、DTLSトンネルの作成に失敗し、認証は失敗します。これは、RADIUSライブログにエラーとして表示されます。Operations > Radius > Live logsの順に移動して確認します。



不明なCAが原因で発生したDTLSハンドシェイクの失敗をISEライブログが報告する

これを修正するには、中間証明書とルート証明書の両方を確認し、Administration>System>Certificates>Trusted certificatesの下のTrust for client authentication and Syslogチェックボックスをオンにします。

失効チェックが実行されている

証明書がWLCにインポートされると、新しく作成されたトラストポイントの失効チェックが有効になります。これにより、WLCは使用できないか到達可能でない証明書失効リスト(CRL)を検索しようとし、証明書検証に失敗します。
証明書検証パスの各トラストポイントに、revocation-check noneコマンドが含まれていることを確認します。

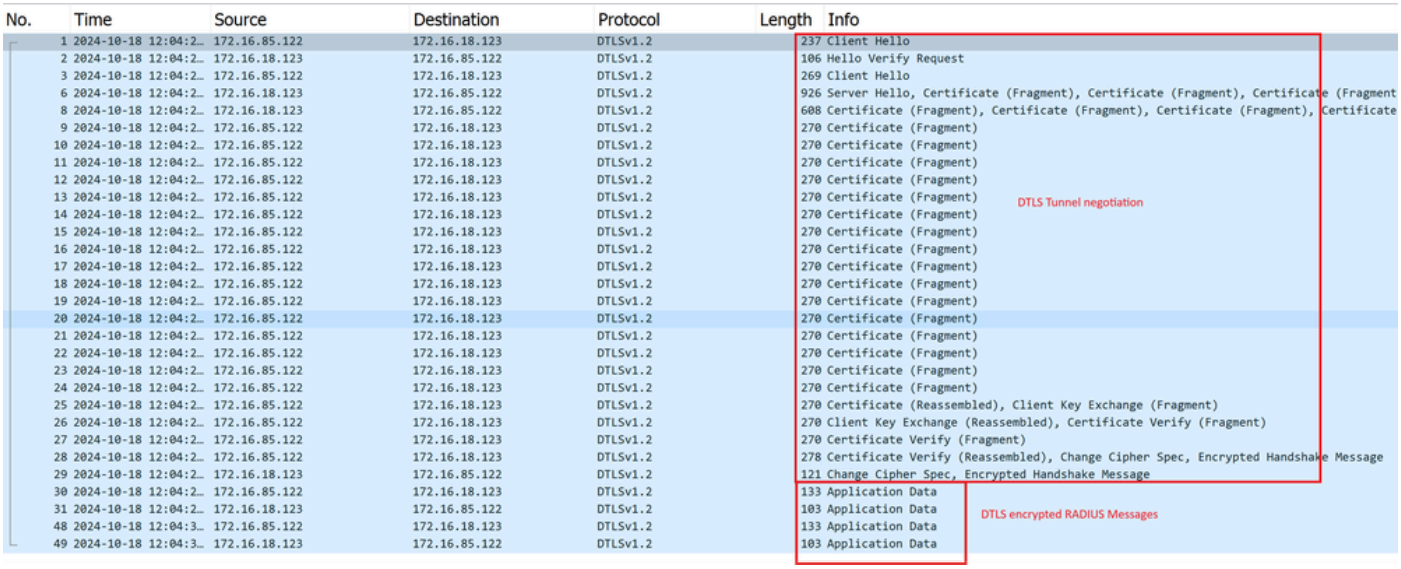
```
Jul 17 21:50:39.064: RADIUS_RADSEC_HASH_KEY_MATCH: hashkey1(0) matches hashkey2(0) TRUE
Jul 17 21:50:39.064: RADIUS_RADSEC_HASH_KEY_GET_CTX: radius radsec  sock_ctx(0x780FB0715978:0) get for
Jul 17 21:50:39.064: RADIUS_RADSEC_PROCESS_SOCK_EVENT: Handle socket event for TLS handshake(172.16.18.
Jul 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
```

パケットキャプチャでのDTLSトンネル確立のトラブルシューティング

9800 WLCには、Embedded Packet Capture(EPC)機能が備わっています。この機能を使用すると、特定のインターフェイスで送受信されたすべてのトラフィックをキャプチャできます。ISEは、着信および発信トラフィックを監視するTCPダンプと呼ばれる同様の機能を提供します。 同時に使用すると、両方のデバイスの観点からDTLSセッション確立トラフィックを分析できます。

ISEでTCPダンプを設定する詳細な手順については、『Cisco Identity Services Engine管理者ガイド』を参照してください。また、WLCでEPC機能を設定する方法については、『Catalyst 9800ワイヤレスLANコントローラのトラブルシューティング』を参照してください。

次に、DTLSトンネルが正常に確立された例を示します。



RADIUS DTLSトンネルネゴシエーションおよび暗号化メッセージのパケットキャプチャ

パケットキャプチャは、DTLSトンネルの確立がどのように行われるかを示します。ネゴシエーションに問題がある場合、デバイス間のトラフィックの損失やDTLS暗号化アラートパケットから、パケットキャプチャは問題を特定するのに役立ちます。

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。