在ISE和9800 WLC上配置Radius DTLS

目录

简介

本文档介绍创建在ISE和9800 WLC之间配置RADIUS DTLS所需的证书的方法。

背景

RADIUS DTLS是RADIUS协议的一种安全形式，其中RADIUS消息通过数据传输层安全(DTLS)隧道发送。要在身份验证服务器和身份验证器之间创建此隧道，需要一组证书。此证书集要求设置某些扩展密钥使用(EKU)证书扩展，具体来说，WLC证书上的客户端身份验证以及ISE证书的服务器身份验证和客户端身份验证。

先决条件

要求

Cisco 建议您了解以下主题：

- 如何配置9800 WLC（接入点[AP]）的基本操作
- 如何使用OpenSSL应用
- 公共密钥基础设施(PKI)和数字证书

使用的组件

本文档中的信息基于以下软件和硬件版本：

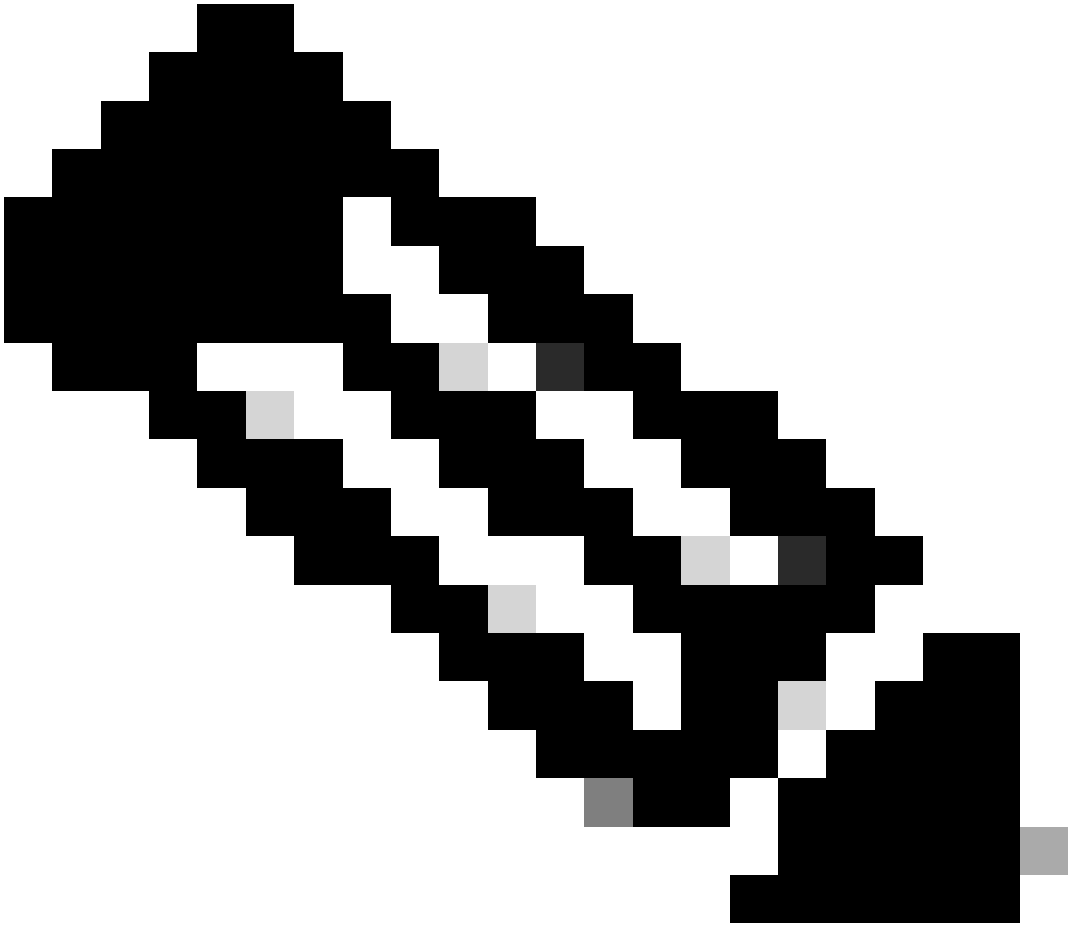
- OpenSSL应用程序（版本3.0.2）。
- ISE（版本3.1.0.518）
- 9800 WLC（版本17.12.3）

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。

配置

概述

目的是创建具有根CA和中间CA的两级证书颁发机构来签署终端证书。证书签名后，会将其导入到WLC和ISE。最后，设备配置为使用这些证书执行RADIUS DTLS身份验证。



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

可选-创建WLC和ISE RADIUS DTLS设备证书

RADIUS DTLS协议需要在ISE和WLC之间交换证书以创建DTLS隧道。如果还没有有效的证书，您可以创建本地CA来生成证书，请参阅在OpenSSL上配置多级证书颁发机构以生成Cisco IOS® XE兼容证书，并从开始到步骤结束时执行文档中概述的步骤 创建中间CA证书。

在openssl.cnf文件中添加配置部分

打开您的openssl.cnf配置文件，并在该文件底部，复制并粘贴用于生成有效证书签名请求(CSR)的WLC和ISE部分。

ISE_device_req_ext和WLC_device_req_ext部分都指向要包含在CSR上的SAN列表：

```
#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
```

作为一项安全措施，CA会覆盖CSR上存在的任何SAN以便对其进行签名，这样未授权设备便无法接收其不允许使用的名称的有效证书。要将SAN重新添加到签名证书中，请使用subjectAltName参数指向与CSR生成所用的SAN相同的列表SAN。

ISE要求证书上同时存在serverAuth和clientAuth EKU，而WLC只需要clientAuth。它们将使用extendedKeyUsage参数添加到签名证书中。

将用于证书签名的部分复制并粘贴到openssl.cnf文件的底部：

```
#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
```

创建WLC设备证书

创建一个新文件夹，用来在计算机上存储WLC证书，该计算机在名为IntermCA.db.certs的中间CA证书文件夹中安装了OpenSSL。新的文件夹称为WLC：

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

修改openssl.cnf文件的[WLC_alt_names]部分中的DNS参数。更改为您所需的值提供的示例名称。这些值填充WLC证书的SAN字段：

```
[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
```

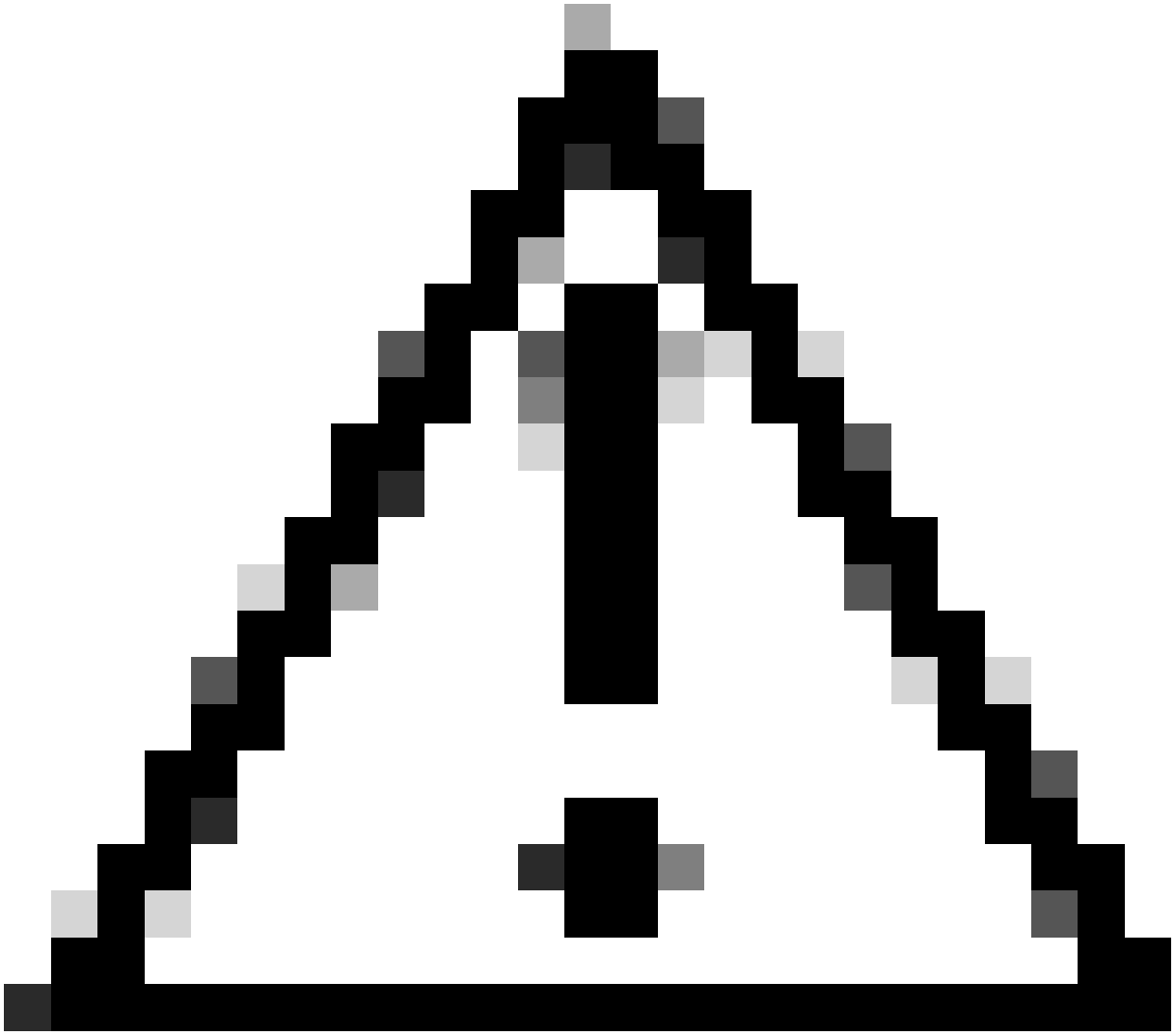
使用WLC_device_req_ext部分中的信息为SAN创建WLC私钥和WLC CSR：

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

OpenSSL会打开一个交互式提示，提示您输入可分辨名称(DN)详细信息：

```
.........+..+.......+...........+...+......+.+...........+..++++++++++++++
++++++
-----
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证书可分辨名称交互式提示



注意：在交互式提示上提供的公用名(CN)必须与openssl.cnf文件的[WLC_alt_names]部分中的名称之一相同。

使用名为IntermCA的CA对名为WLC.csr(扩展名为[WLC_cert])的WLC CSR进行签名，并将签名证书存储在./IntermCA/IntermCA.db.certs/WLC中。WLC设备证书称为WLC.crt：

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

9800 WLC需要使用pfx格式的证书才能将其导入。创建新文件，其中包含签署WLC证书的CA链，这称为certfile：

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

要创建.pfx文件，请根据WLC版本运行以下命令之一。

对于早于17.12.1的版本：

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

对于版本17.12.1或更高版本：

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

创建ISE设备证书

创建一个新文件夹，以便在已将OpenSSL安装在名为IntermCA.db.certs的中间CA证书文件夹中的计算机上存储ISE证书。新文件夹称为ISE：

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

修改openssl.cnf文件的[ISE_alt_names]部分中的DNS参数。更改为您所需的值提供的示例名称，这些值将填充WLC证书的SAN字段：

```
[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
```

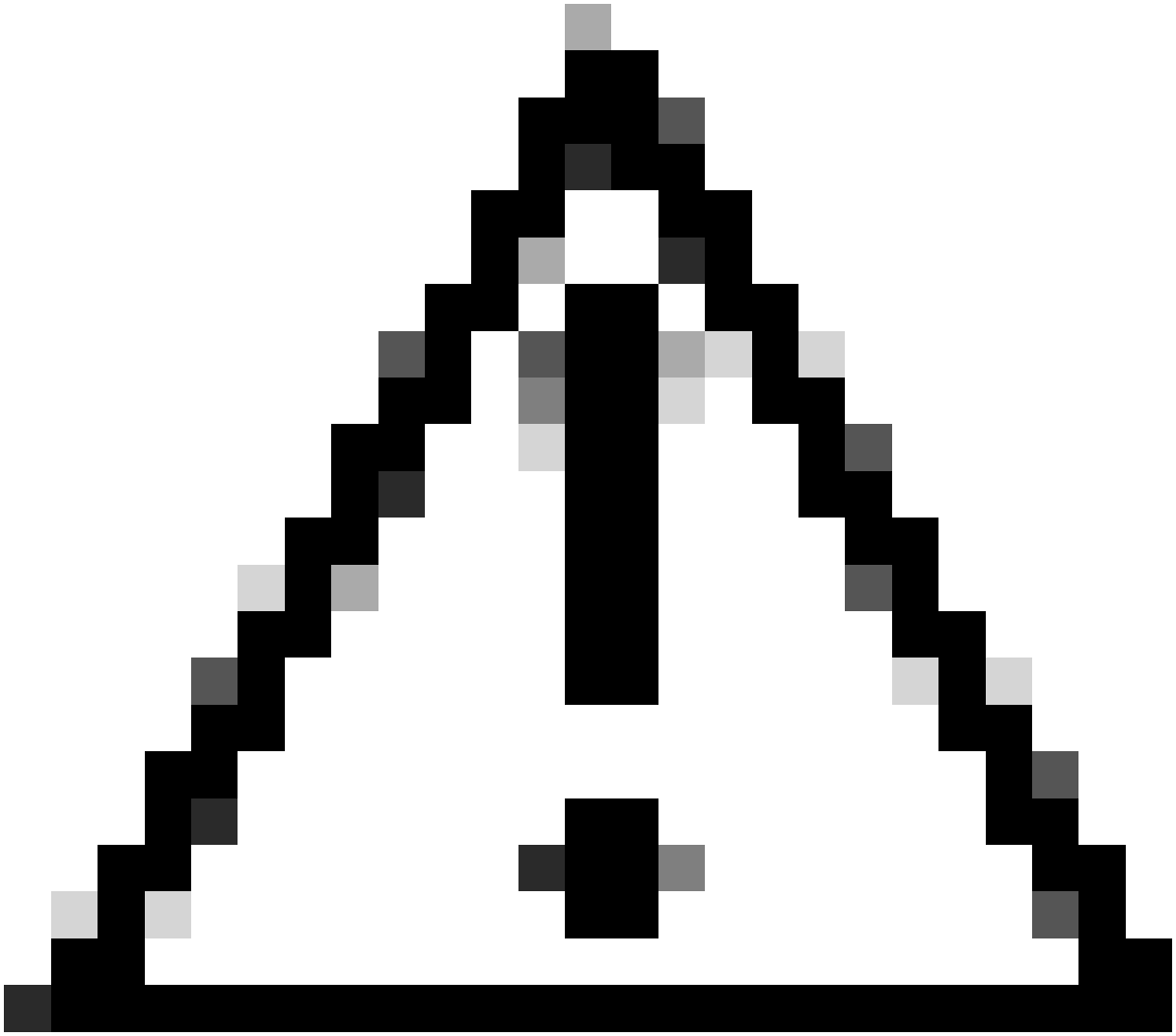
使用ISE_device_req_ext部分中的信息为SAN创建ISE私钥和ISE CSR：

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

OpenSSL会打开一个交互式提示，提示您输入可分辨名称(DN)详细信息：

```
-----
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证书可分辨名称交互式提示



注意：您在交互式提示中提供的CN必须与openssl.cnf文件的[ISE_alt_names]部分中的名称之一完全相同。

使用名为IntermCA的CA对名为ISE.csr(在[ISE_cert] 下定义扩展)的ISE CSR进行签名，并将签名证书存储在./IntermCA/IntermCA.db.certs/WLC中。ISE设备证书称为ISE.crt：

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

将证书导入设备

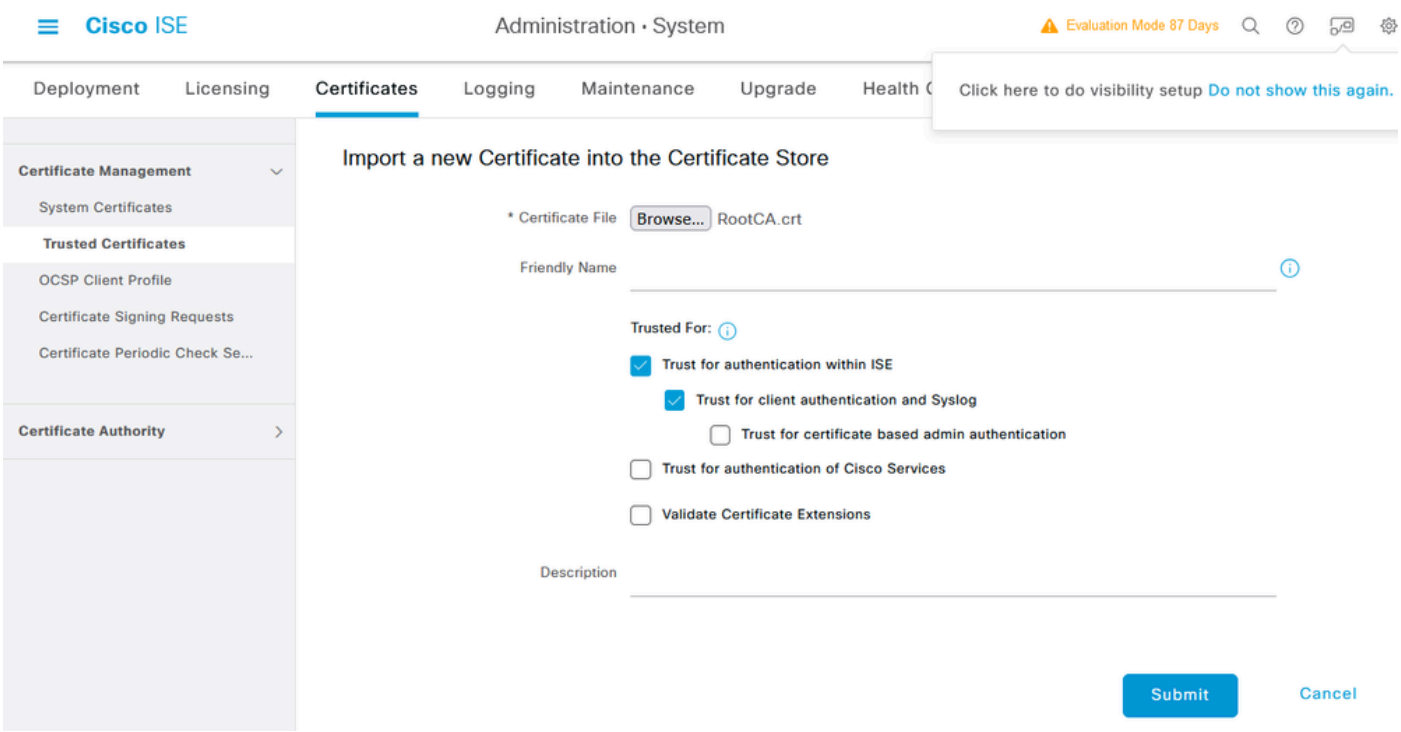
将证书导入ISE

1. 将根CA证书从ISE证书链导入到受信任的证书库中。

2. 导航到管理>系统>证书>受信任证书。

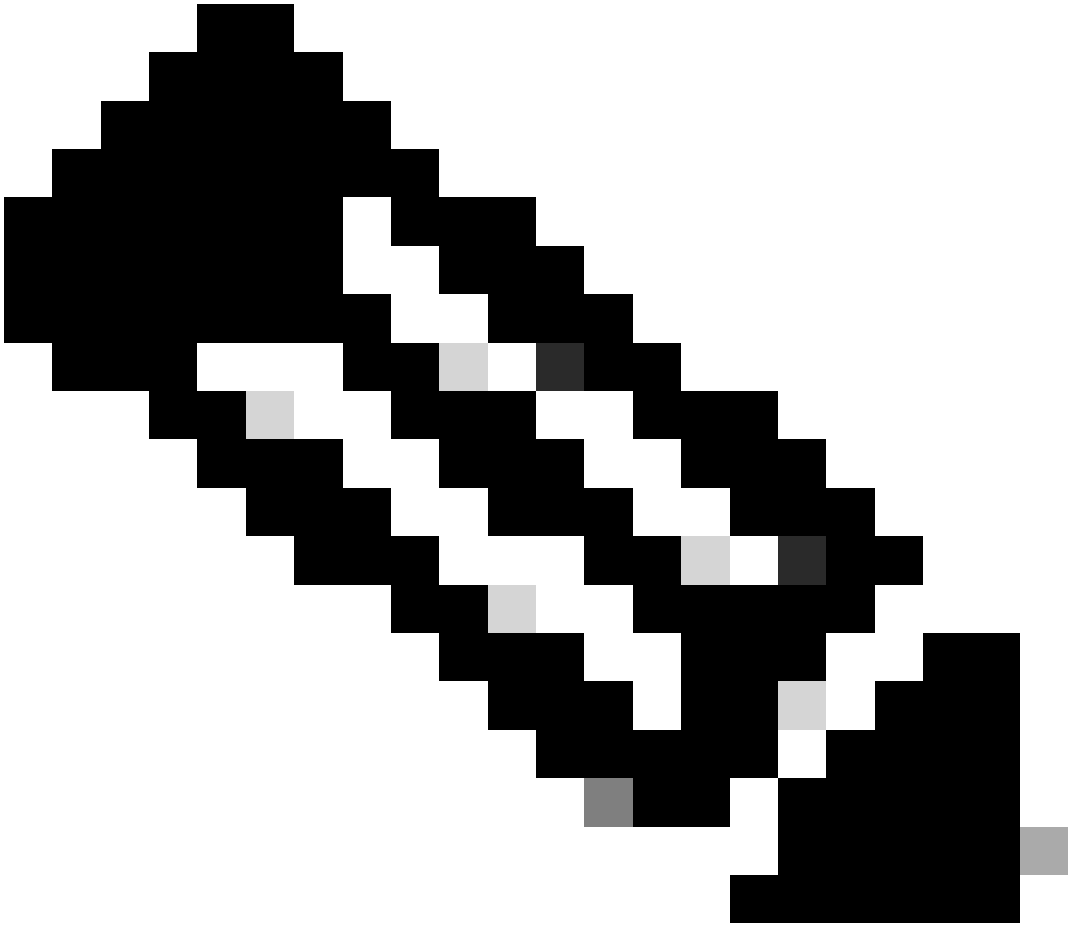
3. 单击“浏览”并选择Root.crt文件。

4. 选中Trust for authentication within ISE 以及Trust for client authentication and Syslog 复选框，然后单击Submit：

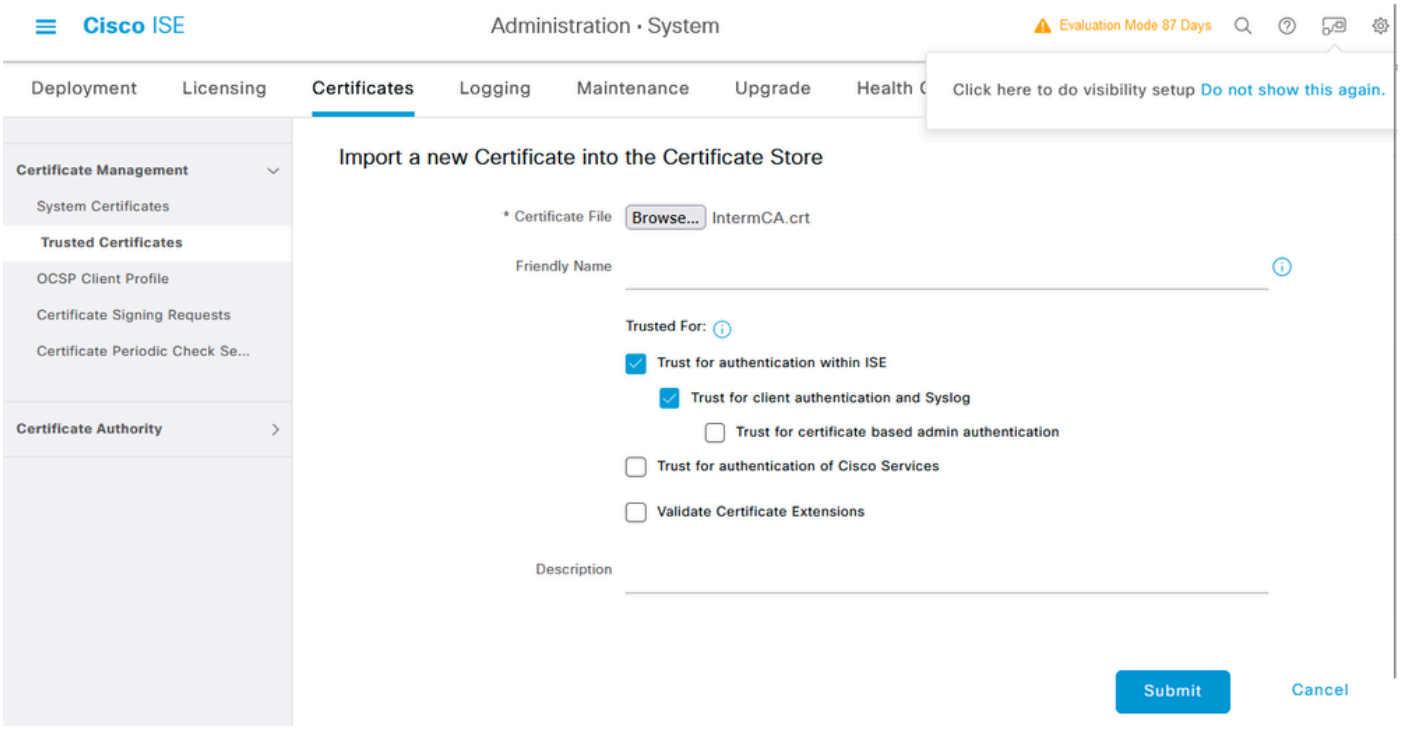


ISE根CA证书导入对话框

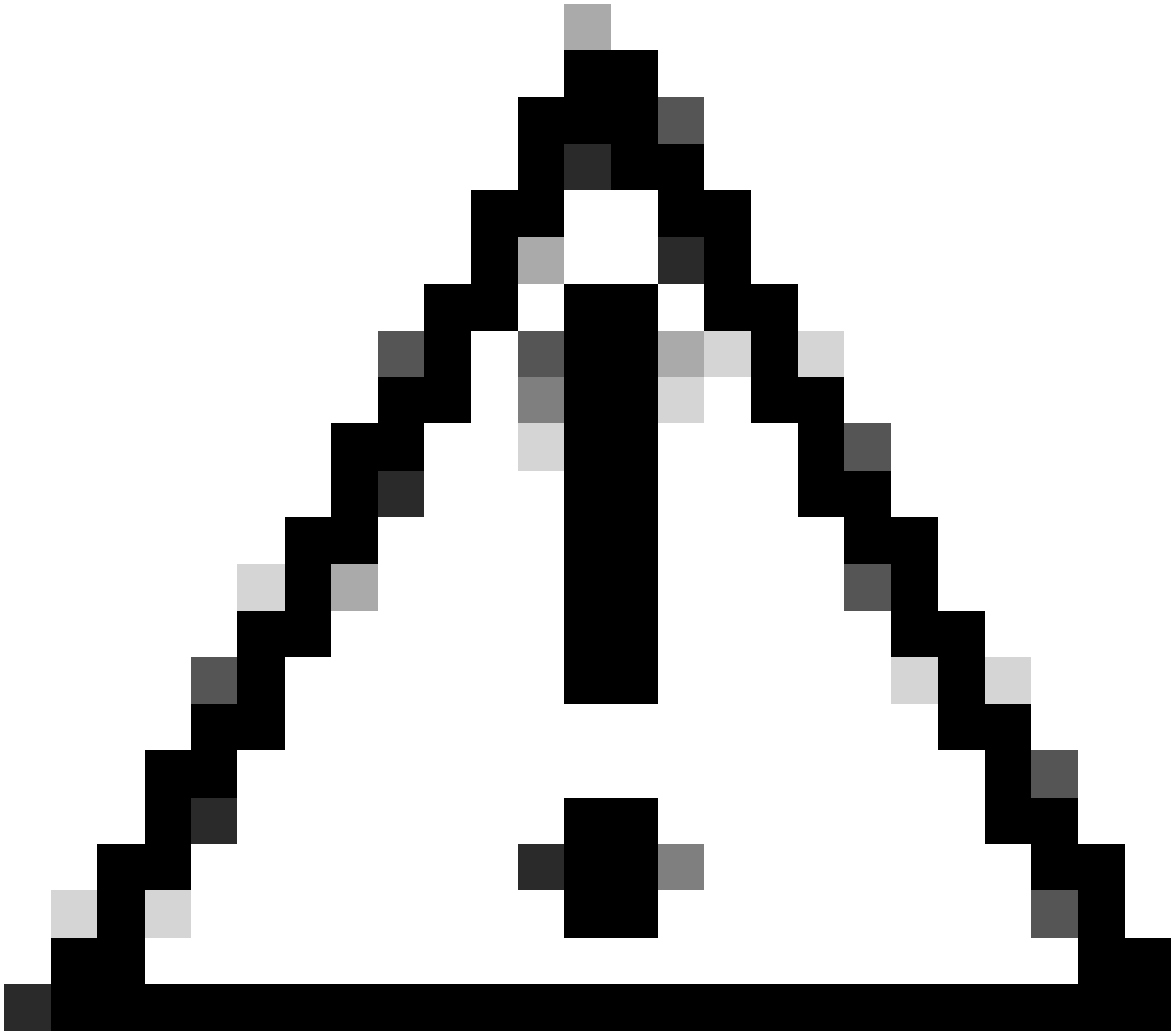
对中间证书（如果存在）执行相同操作。



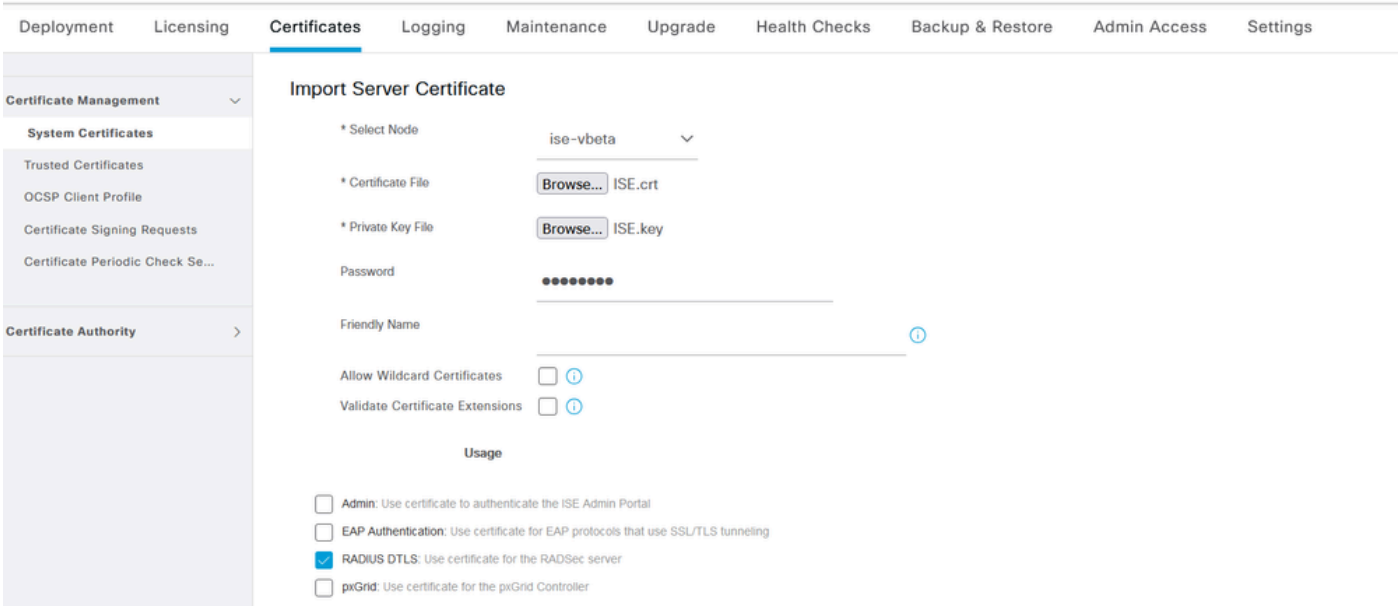
注意：对作为ISE证书验证链一部分的任何CA证书重复上述步骤。始终从根CA证书开始，并以链的最低中间CA证书结束。



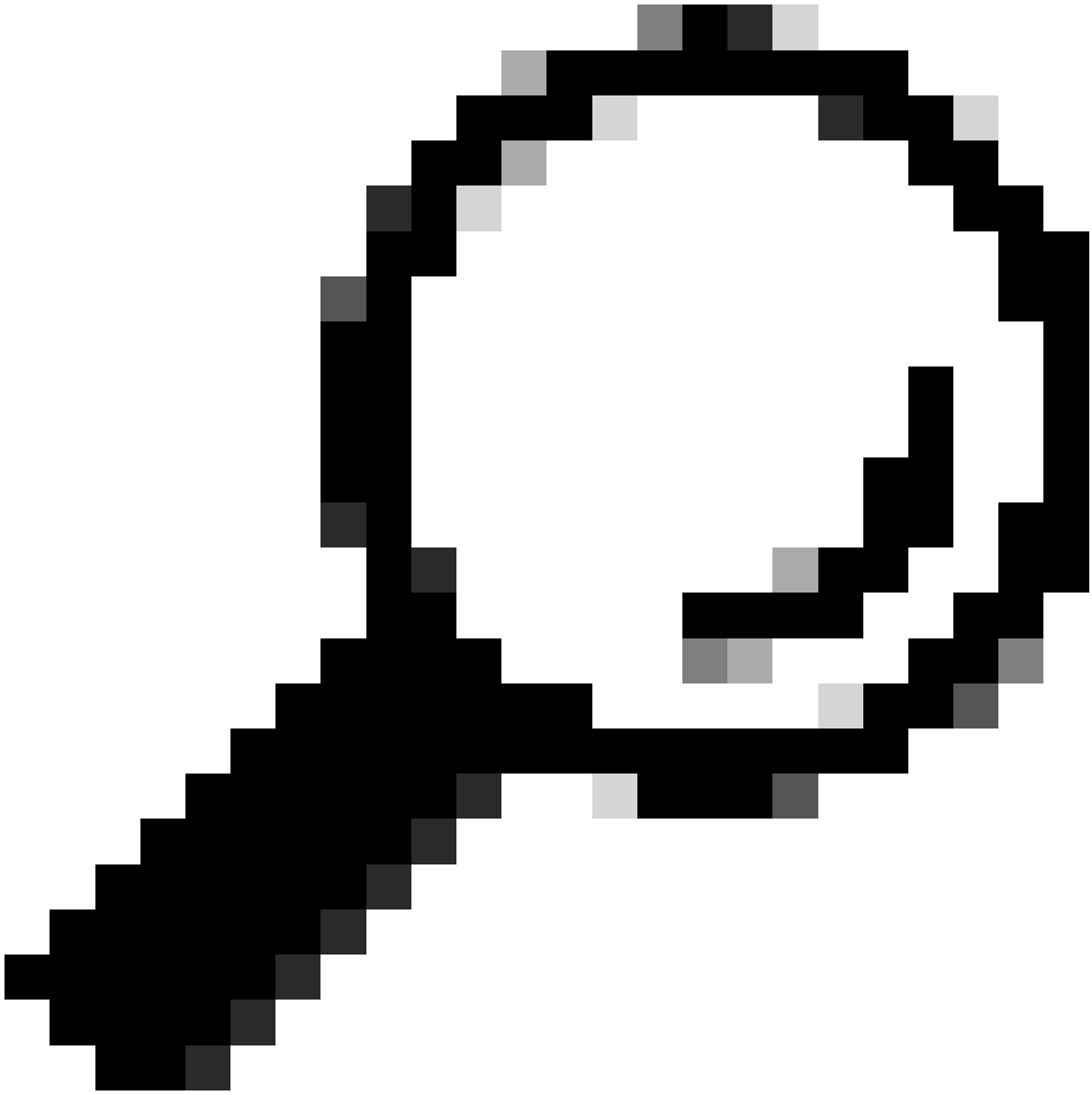
ISE中间CA证书导入对话框



注意：如果ISE证书和WLC证书由不同的CA颁发，则还必须导入属于WLC证书链的所有CA证书。在您导入这些CA证书之前，ISE不会接受DTLS证书交换上的WLC证书。



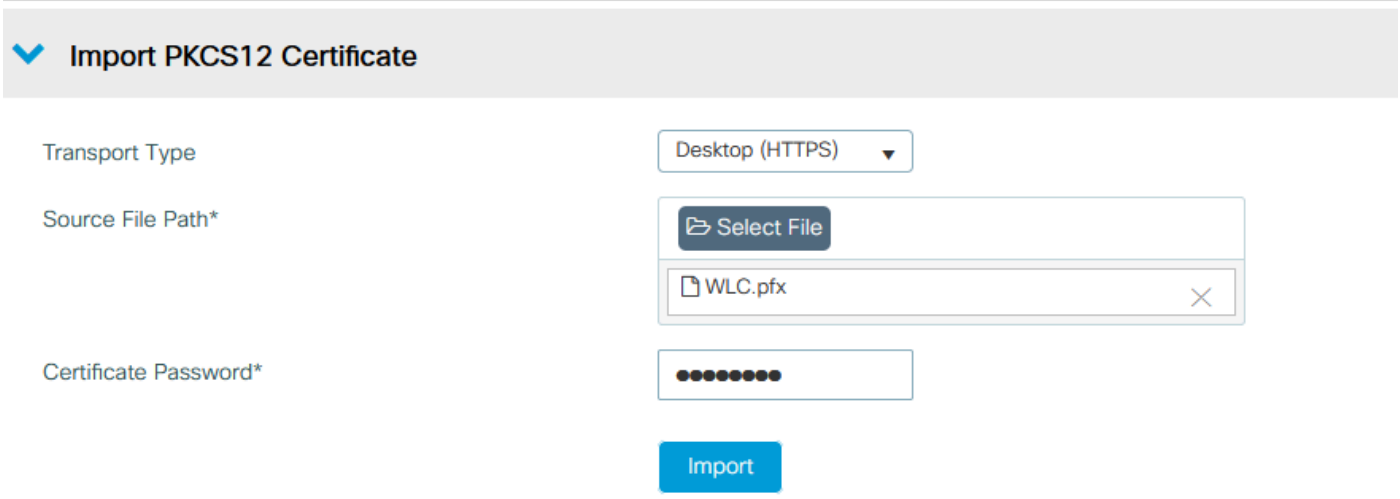
ISE设备证书导入菜单



提示：在此步骤中，您只需导入ISE设备证书。此证书是建立DTLS隧道的ISE交换。无需导入WLC设备证书和私钥，因为将使用以前导入的CA证书验证WLC证书。

将证书导入WLC

1. 导航到WLC上的Configuration > Security > PKI Management，然后转到Add Certificate选项卡。
2. 单击Import PKCS12 Certificate 下拉菜单并将传输类型设置为Desktop (HTTPS)。
3. 单击Select File 按钮，然后选择您之前准备的.pfx文件。
4. 键入导入口令，然后单击Import。

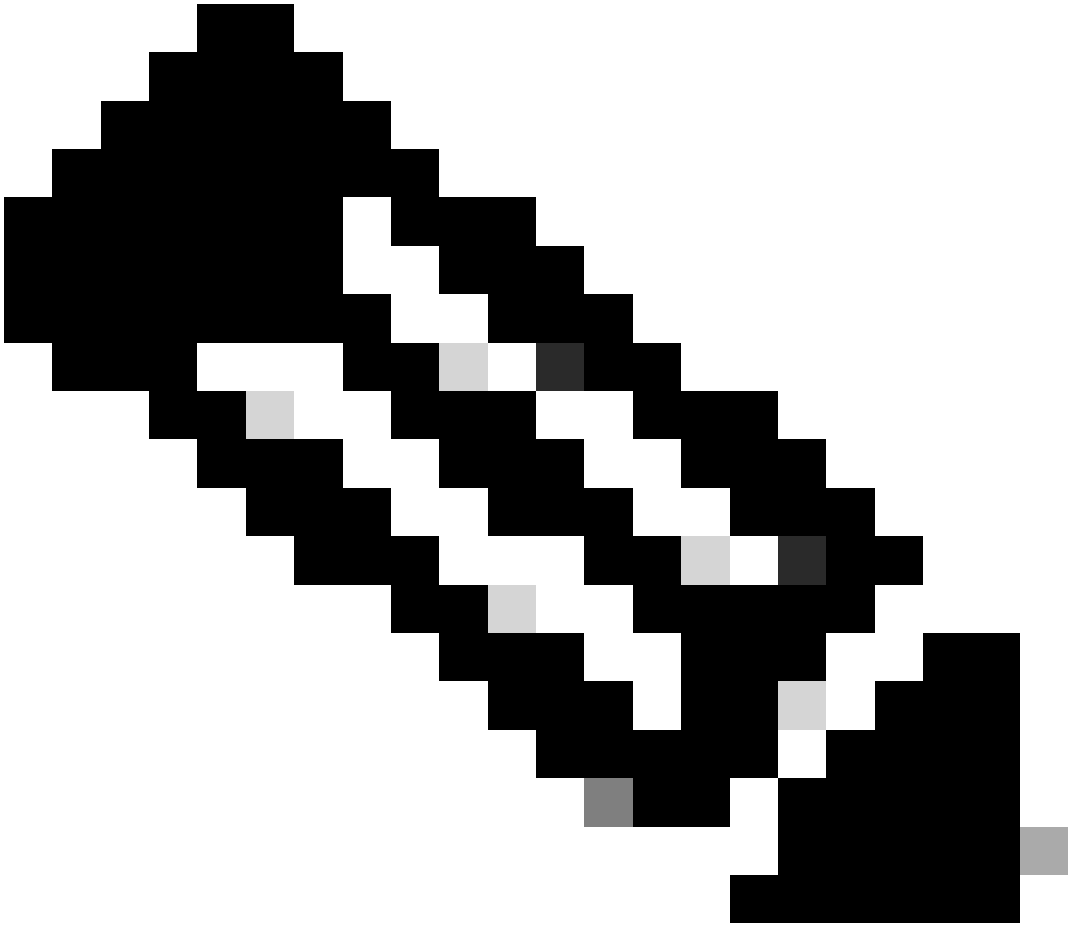


WLC证书导入对话框

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

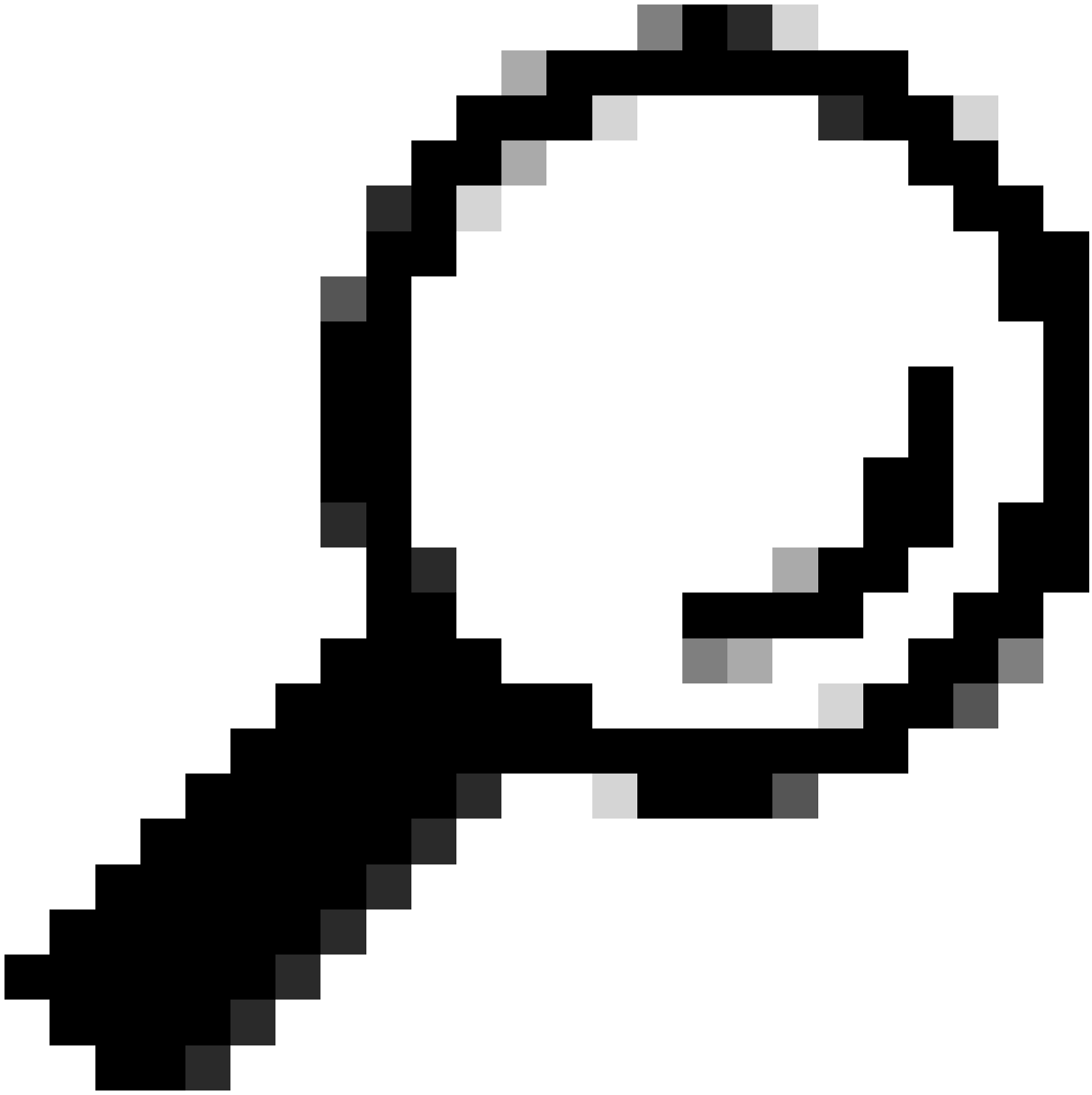
如果WLC没有可通过网络检查的证书撤销列表，请在每个自动创建的信任点内禁用撤销检查：

```
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
```



注意：如果使用在OpenSSL上配置多级CA以生成Cisco IOS XE证书文档在OpenSSL上创建多级CA，则必须禁用撤销检查，因为未创建任何CRL服务器。

自动导入会创建包含WLC证书及其CA证书的必要信任点。



提示：如果WLC证书由与ISE证书相同的CA颁发，您可以使用从WLC证书导入中自动创建的相同信任点。无需单独导入ISE证书。

如果WLC证书由与ISE证书不同的CA颁发，您还需要将ISE CA证书导入WLC，以便WLC信任ISE设备证书。
为根CA创建新的信任点并导入ISE根CA：

```
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-----
```

在ISE CA链上导入下一个中间CA证书，即根CA颁发的CA证书：

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

链中的每个其他CA都需要单独的信任点。链中的每个信任点都必须使用命令chain-validation continue <Issuer trustpoint name>引用包含要导入的证书的颁发者证书的信任点。

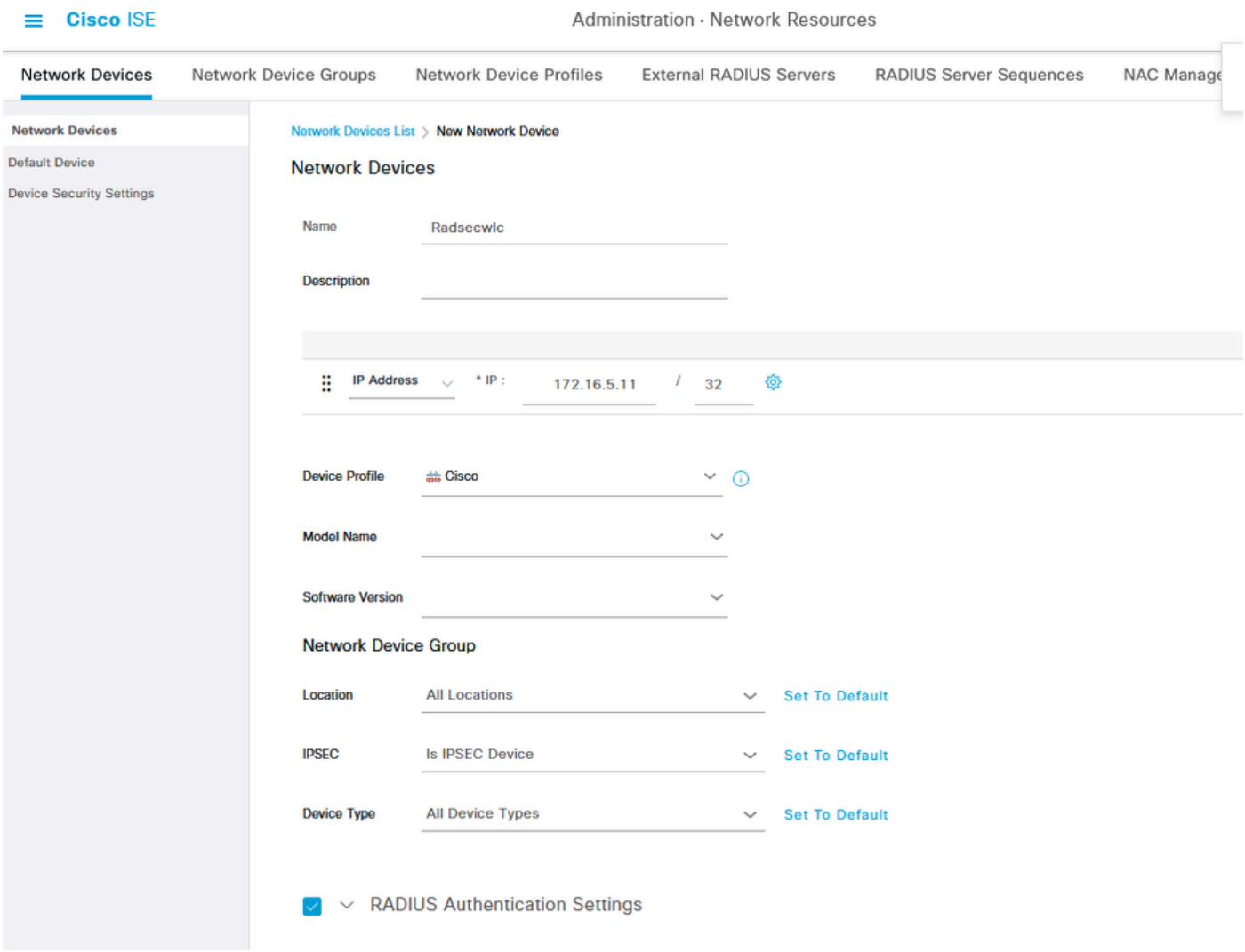
导入与您的CA链包含的CA证书数量相同的证书。在导入ISE设备证书的颁发者CA之后，请记下此信任点的名称。

无需在WLC上导入ISE设备证书，RADIUS DTLS即可工作。

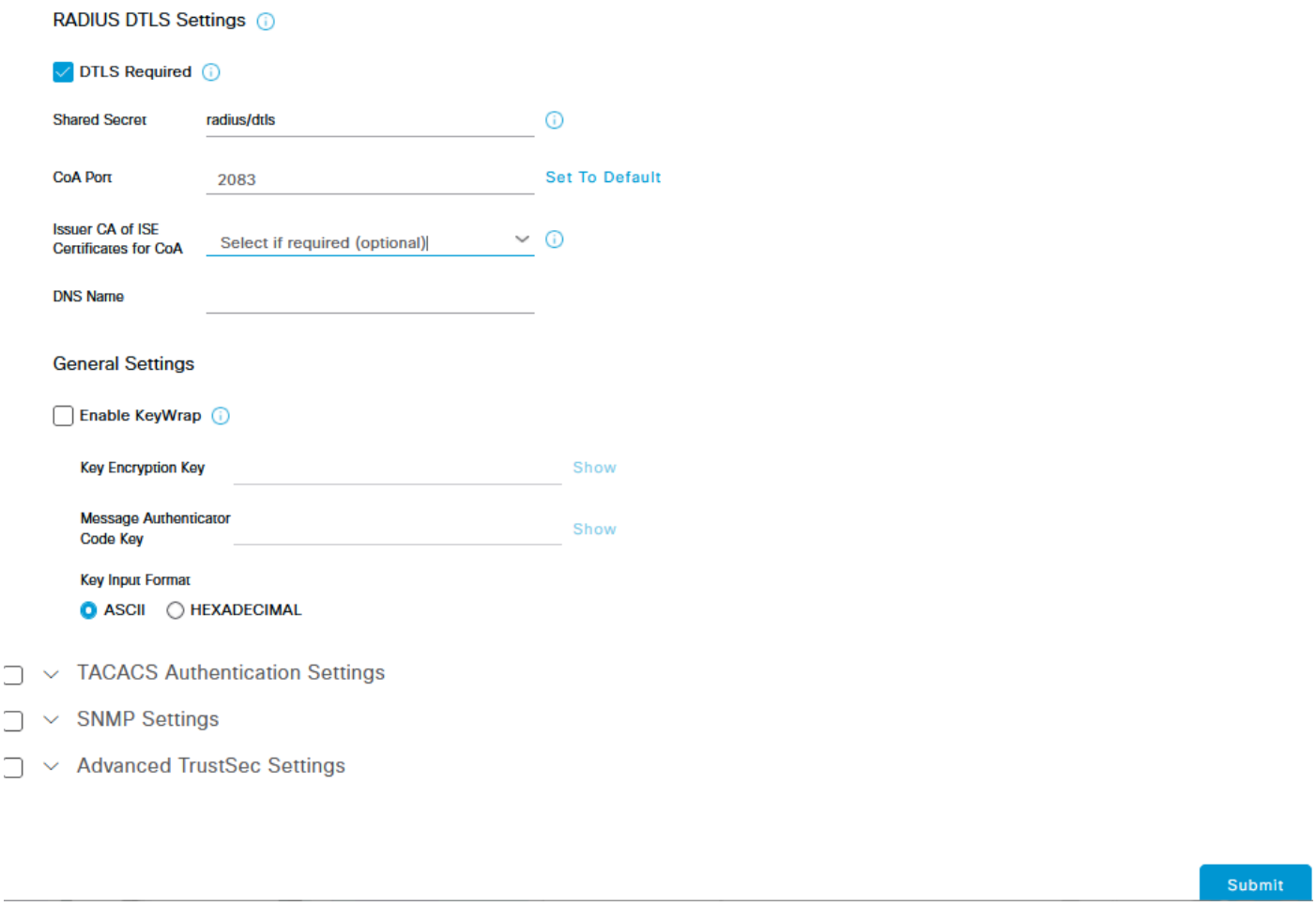
配置RADIUS DTLS

ISE 配置

将WLC作为网络设备添加到ISE中，为此，请导航到管理>网络资源>网络设备>添加
输入设备名称和分配RADIUS流量的WLC接口的IP。通常是无线管理接口IP。向下滚动并选中RADIUS Authentication Settings和DTLS Required，然后单击Submit：


Cisco ISE
Administration · Network Resources
Network Devices
Network Device Groups
Network Device Profiles
External RADIUS Servers
RADIUS Server Sequences
NAC Manage
Network Devices
Default Device
Device Security Settings
Network Devices List > New Network Device
Network Devices
Name
Radsecwlc
Description
IP Address
* IP :
172.16.5.11
/
32
Device Profile
Cisco
Model Name
Software Version
Network Device Group
Location
All Locations
Set To Default
IPSEC
Is IPSEC Device
Set To Default
Device Type
All Device Types
Set To Default
RADIUS Authentication Settings

新网络设备配置



ISE上网络设备的RADIUS DTLS设置

WLC 配置

定义新的RADIUS服务器以及ISE IP地址和RADIUS DTLS的默认端口。此配置仅适用于CLI：

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

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

Radius DTLS必须使用共享密钥radius/dtls，9800 WLC将忽略此密钥以外的任何已配置密钥：

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

使用dtls trustpoint client
命令配置信任点，该信任点包含要交换DTLS隧道的WLC设备证书。
使用dtls trustpoint server
命令配置包含ISE设备证书的颁发者CA的信任点。

只有当WLC和ISE证书由同一CA颁发时，客户端和服务器信任点名称才相同：

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

配置WLC以检查ISE证书上是否存在一个主题备用名称(SAN)。此配置必须与证书的SAN字段中的一个SAN完全匹配。

9800 WLC不为SAN字段执行基于正则表达式的匹配。例如，这意味着通配符证书(其SAN字段中包含*.example.com)的命令 dtls match-server-identity hostname *.example.com“正确”，但通配符证书(其SAN字段中包含www.example.com)的命令不正确。

WLC不会针对任何名称服务器检查此名称：

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

创建新服务器组以使用新的RADIUS DTLS进行身份验证：

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

从此时起，您可以使用此服务器组，就像使用WLC上的任何其他服务器组一样。请参阅在Catalyst 9800无线控制器系列上配置802.1X身份验证，以将此服务器用于无线客户端身份验证。

验证

验证证书信息

要验证已创建证书的证书信息，请在Linux终端上运行命令：

```
openssl x509 -in

      -text -noout
```

显示完整的证书信息。这对于确定给定证书的颁发者CA或证书是否包含所需的EKU和SAN非常有用：

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

OpenSSL所示的Cisco IOS XE设备证书信息

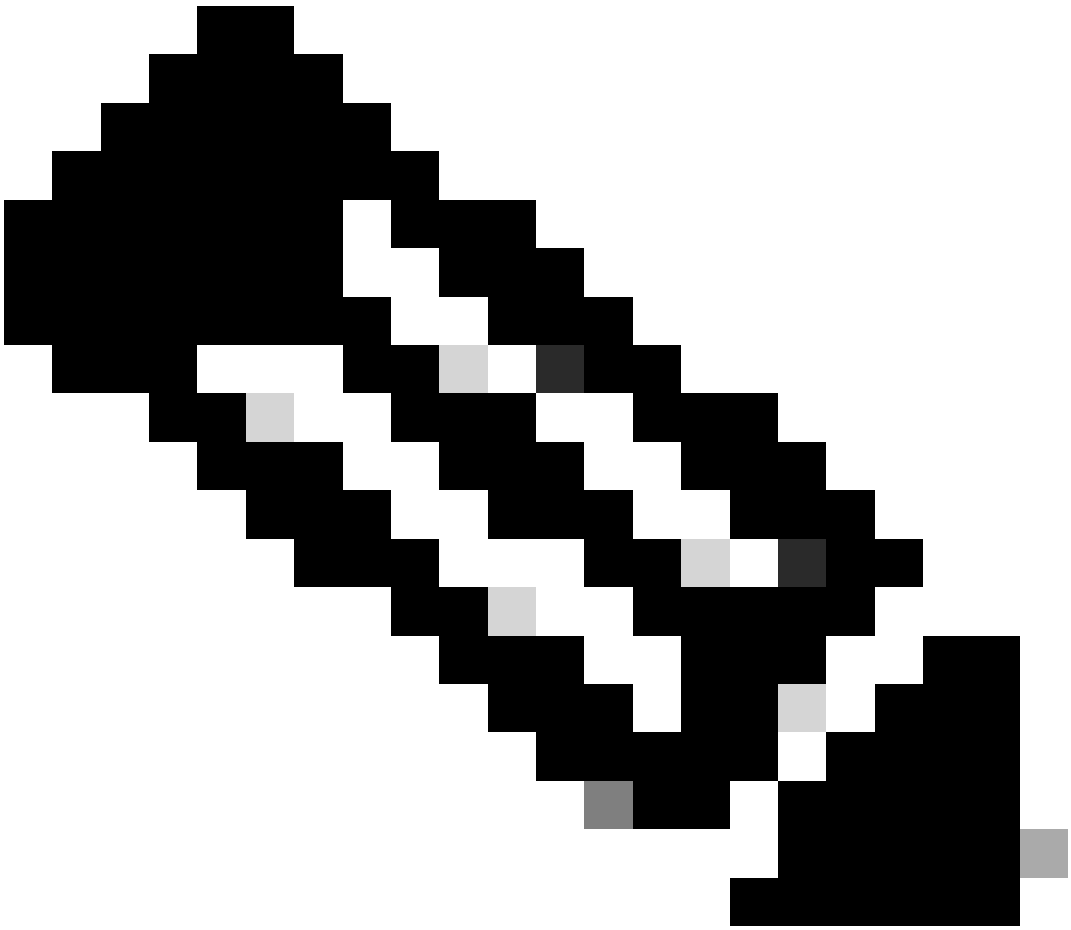
执行测试身份验证

从WLC中，您可以使用命令测试Radius DTLS功能 test aaa group

new-code

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

```
USER ATTRIBUTES

username             0   "testuser"
```



注意：test命令上的访问拒绝输出表示WLC收到Access-Reject RADIUS消息，在这种情况下RADIUS DTLS正常工作。但是，这也可能表示无法建立DTLS隧道。test命令无法区分两种情况，请参阅故障排除部分，以确定是否存在问题。

故障排除

要查看身份验证失败的原因，可以在执行测试身份验证之前启用这些命令。

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

以下是在启用调试的情况下成功进行身份验证的输出：

```
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报告的未知CA

当WLC无法验证ISE提供的证书时，无法创建DTLS隧道，并且身份验证失败。

以下是出现这种情况时显示的调试消息的示例：

```
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:
```

要更正此问题，请确保WLC上配置的身份与ISE证书中包含的其中一个SAN完全匹配：

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

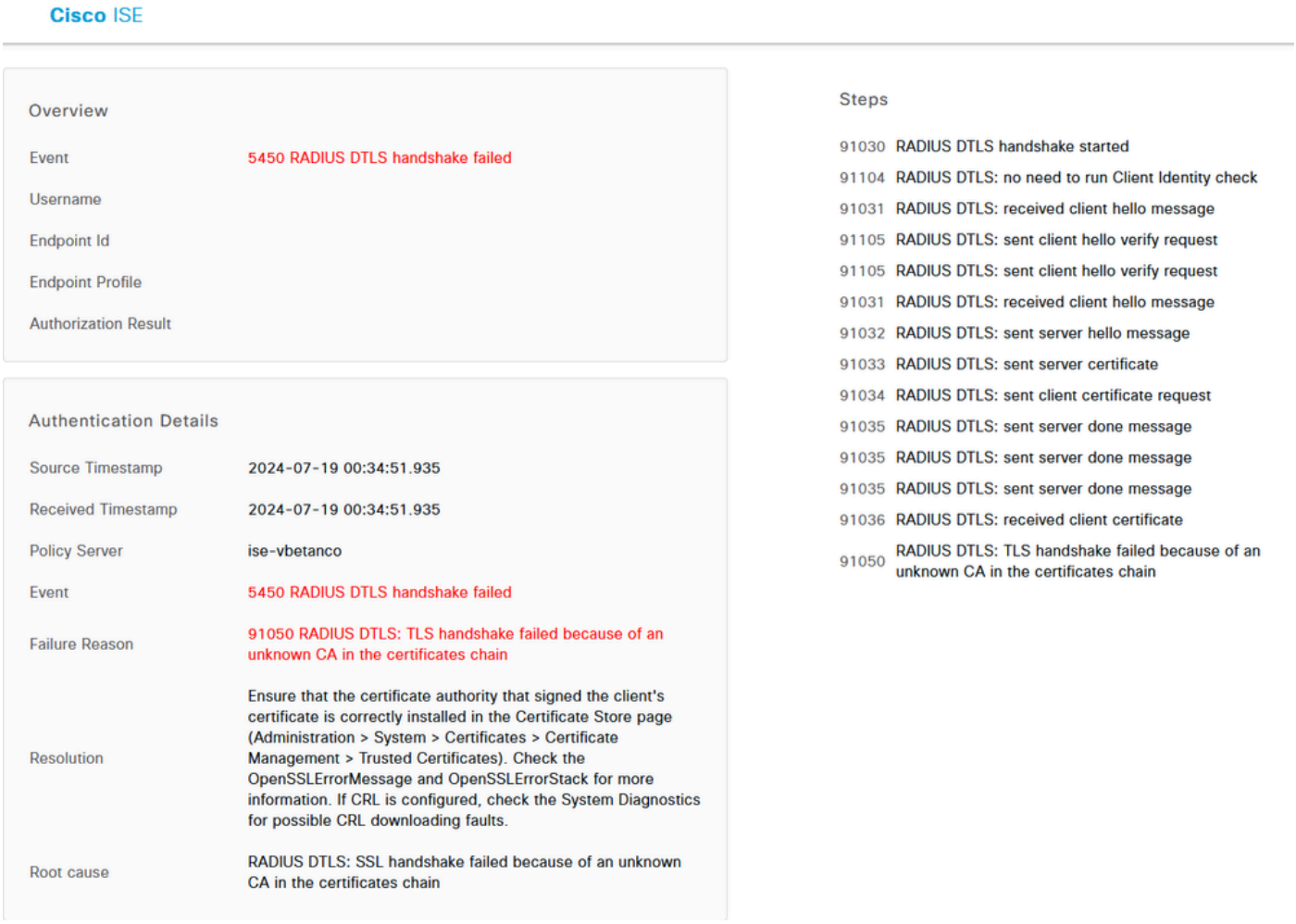
```
      9800(config)#dtls match-server-identity hostname
```

确保已在控制器上正确导入CA证书链，并且 **dtls trustpoint server**

configuration uses the Issuer CA trustpoint.

ISE报告的未知CA

当ISE无法验证WLC提供的证书时，它无法创建DTLS隧道，并且身份验证失败。这在RADIUS实时日志中显示为错误。导航到操作>Radius>实时日志进行验证。



ISE实时日志报告由于未知CA导致的DTLS握手失败

要更正该错误，请确保中间证书和根证书，然后在Administration>System>Certificates>Trusted certificates下选中Trust for client authentication和Syslog复选框。

已进行吊销检查

当证书导入到WLC时，新创建的信任点启用撤销检查。这会使WLC尝试搜索不可用或可访问的证书撤销列表，并且证书验证失败。
确保证书的验证路径中的每个信任点都包含命令revocation-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 c
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)
```

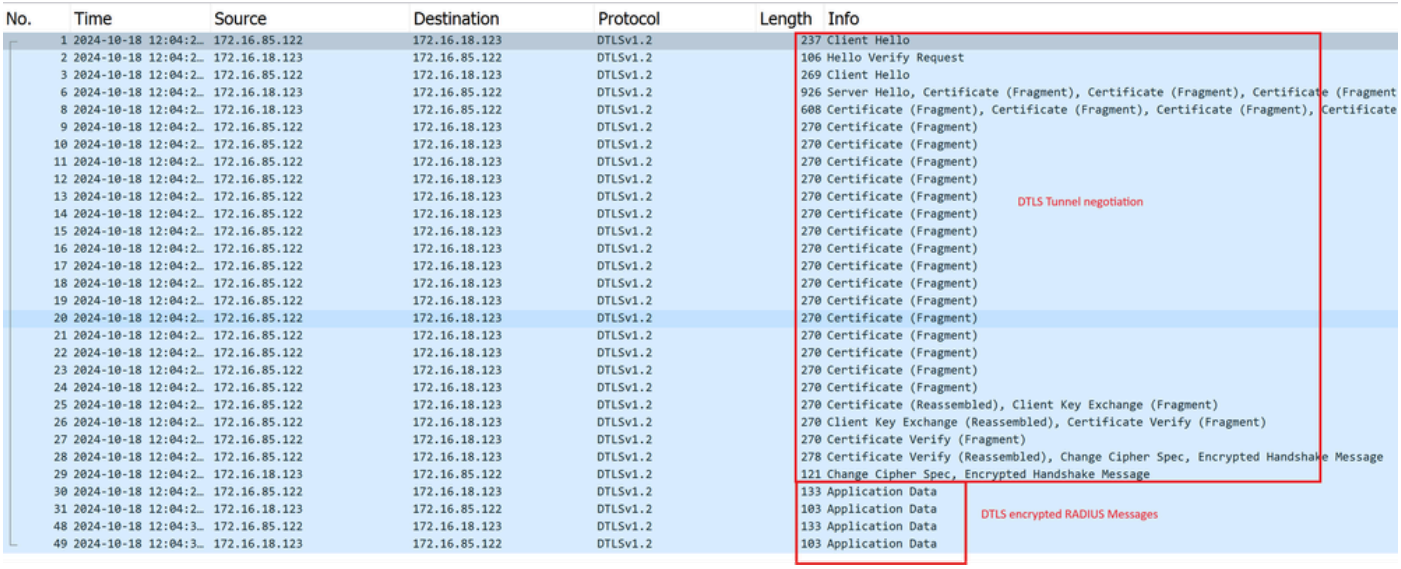
```
Jul 17 21:50:39.070: RADIUS_RADSEC_CONN_CLOSE: Cleaned up timers for Radius RADSEC ctx
Jul 17 21:50:39.070: RADIUS_RADSEC_GENERATE_HASHKEY: hash key(0) generated for sock(0)
Jul 17 21:50:39.070: RADIUS_RADSEC_GENERATE_HASHBUCKET: hash bucket(0) generated for sock(0)
Jul 17 21:50:39.070: RADIUS_RADSEC_HASH_KEY_DEL_CTX: 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
```

对数据包捕获上的DTLS隧道建立进行故障排除

9800 WLC提供嵌入式数据包捕获(EPC)功能，通过该功能，您可以捕获给定接口的所有已发送和已接收流量。ISE提供称为TCP转储的类似功能来监控传入和传出流量。 同时使用时，它们允许您从两个设备的角度分析DTLS会话建立流量。

有关在ISE上配置TCP转储的详细步骤，请参阅思科身份服务引擎管理员指南。另请参阅Catalyst 9800无线LAN控制器故障排除，了解有关如何配置WLC上的EPC功能的信息。

这是成功建立DTLS隧道的示例。



RADIUS DTLS隧道协商和加密邮件的数据包捕获

数据包捕获显示如何建立DTLS隧道。如果协商出现问题（从设备之间的流量丢失或DTLS加密的警报数据包丢失），数据包捕获可帮助您识别问题。

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

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

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