Troubleshoot Expressway Traffic Server Certificate Verification for MRA Services Introduced by CSCwc69661 / CSCwa25108

Contents

Introduction

This document describes the behavior change on Expressway versions of X14.2.0 and higher linked to Cisco bug ID CSCwc69661 or Cisco bug ID CSCwa25108.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Expressway basic configuration
- MRA basic configuration

Components Used

The information in this document is based on Cisco Expressway on version X14.2 and higher.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

With this change of behavior marked by Cisco bug ID CSCwc69661 or Cisco bug ID CSCwa25108 , the traffic server on the Expressway platform performs certificate verification of the Cisco Unified Communication Manager (CUCM), Cisco Unified Instant Messaging & Presence (IM&P) and Unity server nodes for the Mobile and Remote Access (MRA) services. This change can lead to MRA login failures after an upgrade on your Expressway platform.

Hypertext Transfer Protocol Secure (HTTPS) is a secure communication protocol which uses Transport Layer Security (TLS) to encrypt the communication. It does create this secure channel by the use of a TLS certificate that is exchanged in the TLS handshake. In that way, it servers two purposes: authentication (to know who the remote party is you connect to) and privacy (the encryption). The authentication protects against man-in-the-middle attacks and the privacy prevents attackers to eavesdrop and tamper on the communication.

TLS (certificate) verification is performed in the sight of authentication and allows you to be sure that you have connected to the right remote party. The verification consists of two individual items:

1. Trusted Certificate Authority (CA) chain

2. Subject Alternative Name (SAN) or Common Name (CN)

Trusted CA Chain

In order for Expressway-C to trust the certificate that CUCM / IM&P / Unity sends, it needs to be able to establish a link from that certificate to a top level (root) Certification Authority (CA) that it trusts. Such a link, a hierarchy of certificates that link an entities certificate to a root CA certificate, is called a chain of trust. To be able to verify such a chain of trust, each certificate contains two fields : Issuer (or 'Issued by') and Subject (or 'Issued To').

Server certificates, such as the one CUCM sends to Expressway-C, have in the 'Subject' field typically their Fully Qualified Domain Name (FQDN) in the CN:

```
Issuer: DC=lab, DC=vngtp, CN=vngtp-ACTIVE-DIR-CA
Subject: C=BE, ST=Flamish-Brabant, L=Diegem, O=Cisco, OU=TAC, CN=cucm.vngtp.lab
```

Example of a server certificate for CUCM cucm.vngtp.lab. It has the FQDN in the CN attribute of the Subject field together with other attributes such as the Country (C), State (ST), Location (L), ... We can see also that the server certificate is handed out (issued) by a CA called vngtp-ACTIVE-DIR-CA.

Top level CAs (root CAs) can also issue a certificate to identify themselves. In such root CA certificate, we see that the Issuer and Subject have the same value :

```
Issuer:  DC=lab, DC=vngtp, CN=vngtp-ACTIVE-DIR-CA
Subject: DC=lab, DC=vngtp, CN=vngtp-ACTIVE-DIR-CA
```

It is a certificate handed out by a root CA to identify itself.

In a typical situation, root CAs do not directly issue server certificates. Instead, they issue certificates for other CAs. Such other CAs are then called intermediate CAs. Intermediate CAs can in turn directly issue server certificates or certificates for other intermediate CAs. We can have a

situation where a server certificate is issued by intermediate CA 1, which in turn gets a certificate from intermediate CA 2 and so on. Until finally intermediate CA gets its certificate straight from the root CA :

```
Server certificate :
Issuer: DC=lab, DC=vngtp, CN=vngtp-intermediate-CA-1 Subject: C=BE, ST=Flamish-Brabant,
L=Diegem, O=Cisco, OU=TAC, CN=cucm.vngtp.lab
Intermediate CA 1 certificate :
Issuer: DC=lab, DC=vngtp, CN=vngtp-intermediate-CA-2
Subject: DC=lab, DC=vngtp, CN=vngtp-intermediate-CA-1
Intermediate CA 2 certificate :
Issuer: DC=lab, DC=vngtp, CN=vngtp-intermediate-CA-3
Subject: DC=lab, DC=vngtp, CN=vngtp-intermediate-CA-2
...
Intermediate CA n certificate :
Issuer: DC=lab, DC=vngtp, CN=vngtp-ACTIVE-DIR-CA
Subject: DC=lab, DC=vngtp, CN=vngtp-intermediate-CA-n
Root CA certificate :
Issuer: DC=lab, DC=vngtp, CN=vngtp-ACTIVE-DIR-CA
Subject: DC=lab, DC=vngtp, CN=vngtp-ACTIVE-DIR-C
```

Now, in order for Expressway-C to trust the server certificate that CUCM sends, it needs to be able to build the chain of trust from that server certificate up until a root CA certificate. For that to happen, we need to upload the root CA certificate and also all the intermediate CA certificates (if there are any, which is not the case if the root CA would have directly issued the server certificate of CUCM) in the trust store of Expressway-C.

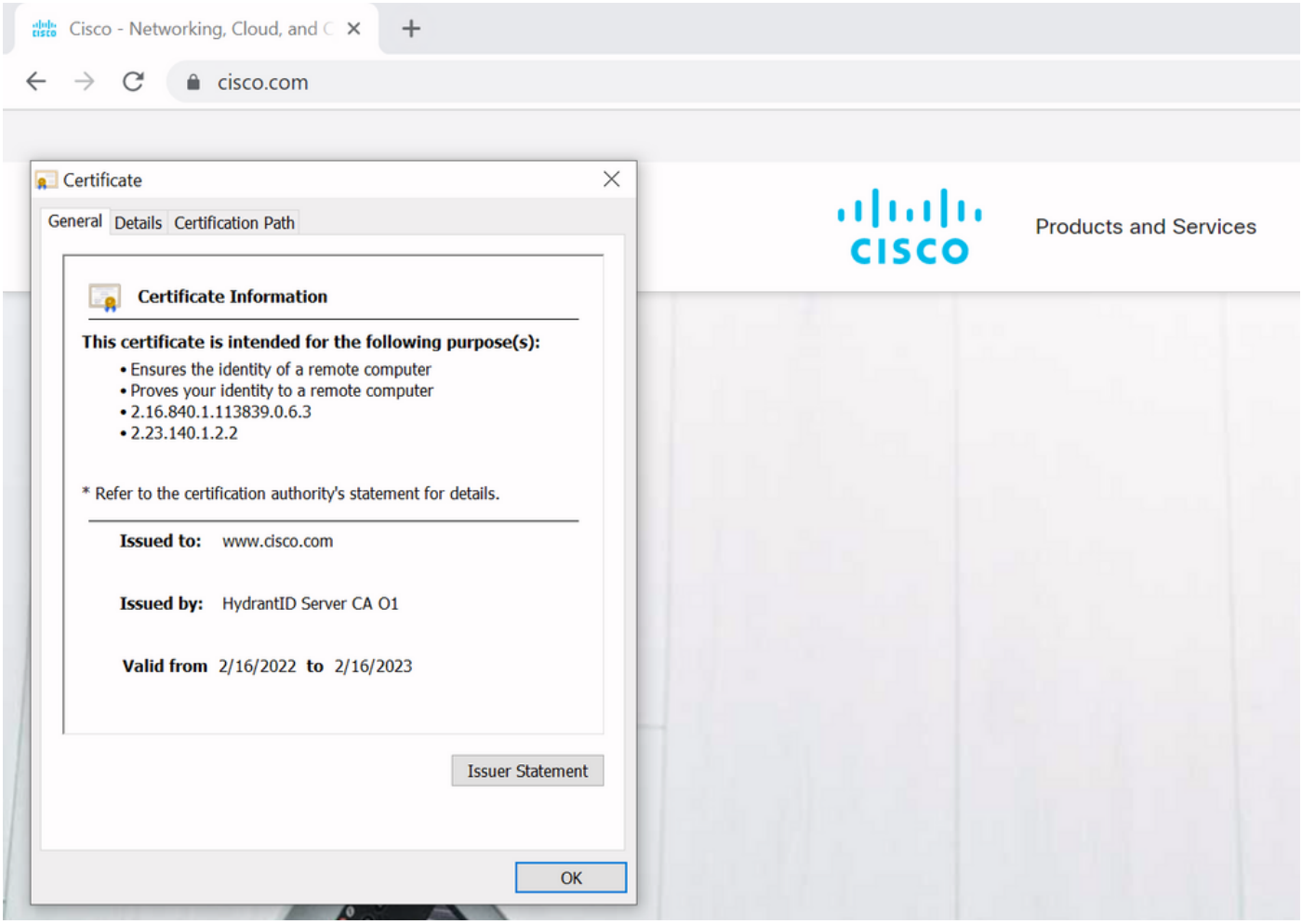
> **Note: Although the Issuer and Subject fields are easy to build the chain of Trust in a human readable way, CUCM does not use these fields in the certificate. Instead, it uses the 'X509v3 Authority Key Identifier' and 'X509v3 Subject Key Identifier' fields to build the chain of trust. Those keys contain identifiers for the certificates which are more accurate then to use the Subject/Issuer fields : there can be 2 certificates with the same Subject/Issuer fields but one of them is expired and one is still valid. They would both have a different X509v3 Subject Key identifier so CUCM can still determine the correct chain of trust.**
>
> **This is not the case for Expressway though as per Cisco bug ID CSCwa12905 and it is not possible to upload two different (self-signed for example) certificates into the trust store of Expressway that have the same Common Name (CN). The way to correct on this, is to CA signed certificates or to use different Common Names for it or to see that it uses always the same certificate (potentially through the re-use certificate feature in CUCM 14).**

SAN or CN Check

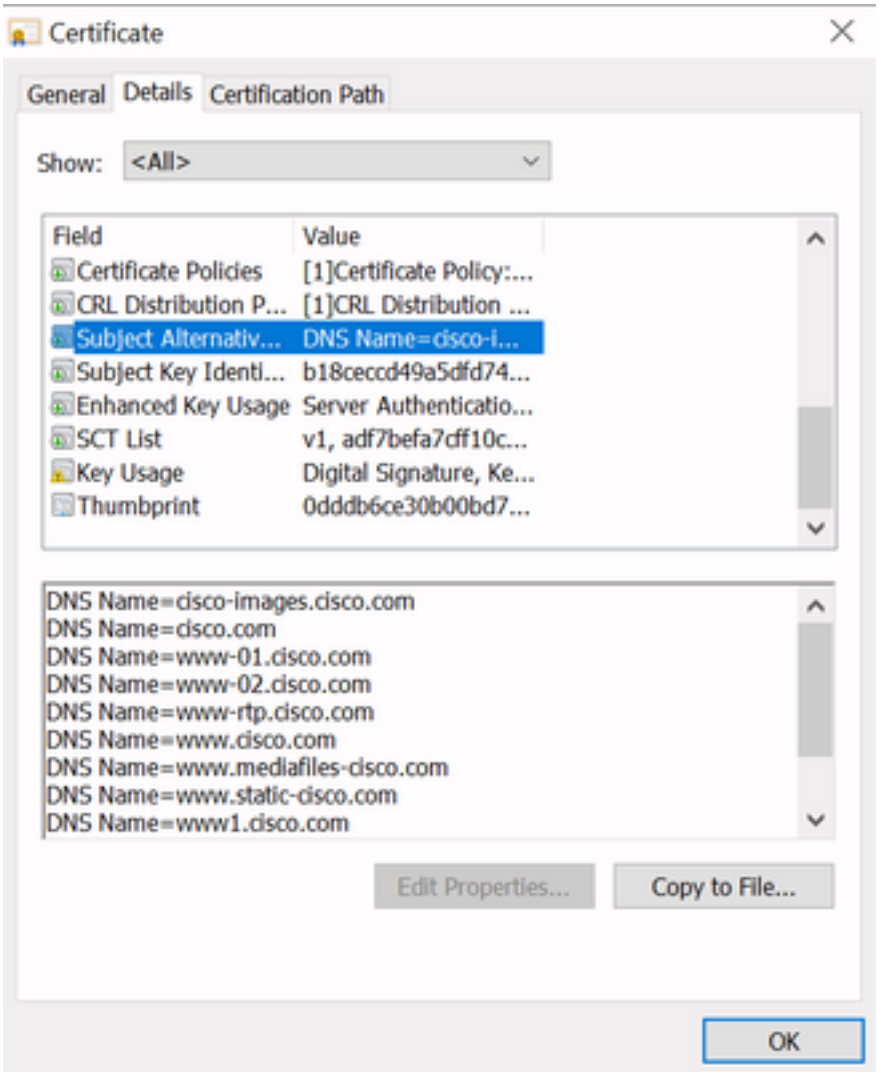
Step 1 checks out the trust store, however anyone who has a certificate that was signed by a CA in the trust store would be valid then. This clearly is not sufficient. Therefore, there is an additional check that validates that the server that you connect to specifically is indeed the correct one. It does this based on the address for which the request was made.

The same kind of operation happens in your browser so let us look into this through an example. If you browse to https://www.cisco.com you see a lock icon next to the URL you entered and it means that it is a trusted connection. This is based both on the CA trust chain (from first section) as well as on the SAN or CN check. If we open up the certificate (via the browser by a click on the

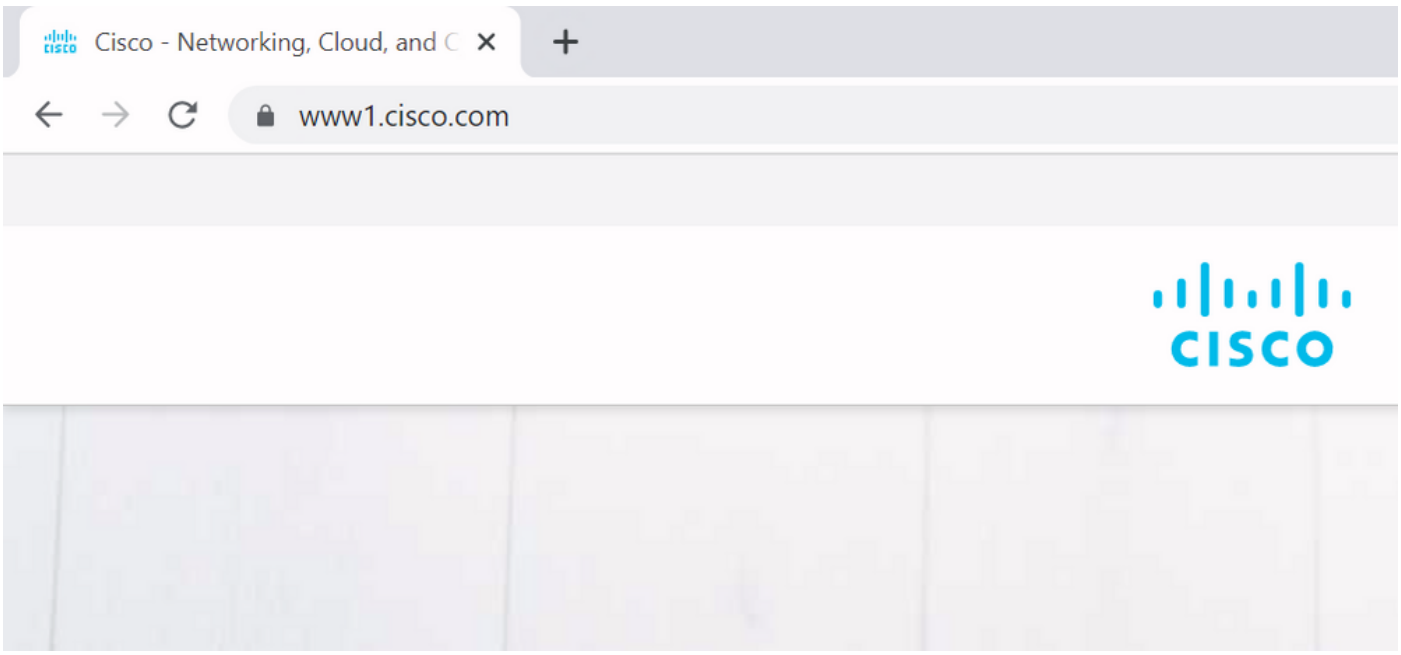
lock icon), you see that the Common Name (seen on 'Issued to:' field) is set to www.cisco.com and that corresponds exactly to the address that we wanted to connect to. In that way it can be sure that we connect to the right server (because we trust the CA who signed the certificate and which performs verification before it hands out the certificate).



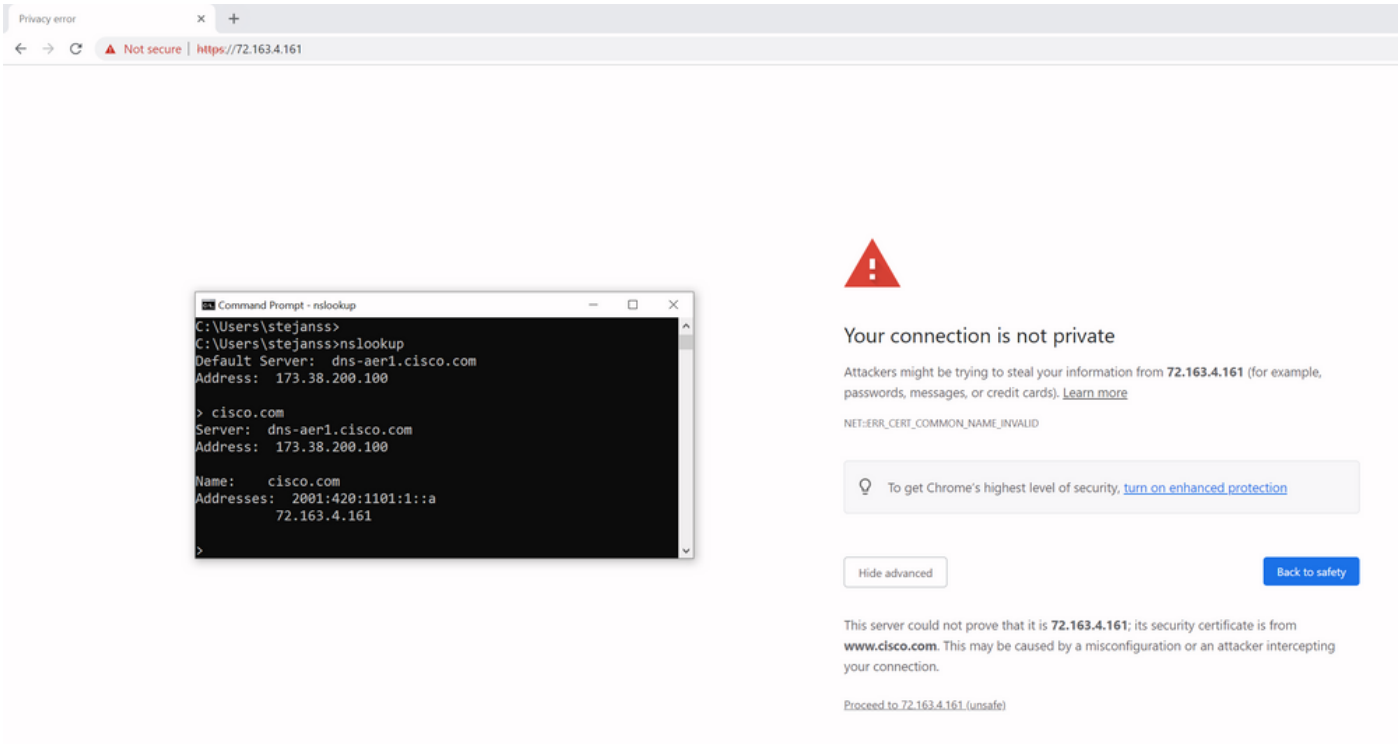
When we look to the details of the certificate and in particular to the SAN entries, we see that the same is repeated as well as some other FQDNs:



This means that when we would request to connect to https://www1.cisco.com for example, that it would show up as a secure connection as well because it is contained in the SAN entries.



However when we would not browse to https://www.cisco.com but rather directly to the IP address (https://72.163.4.161) then it does not show up a secure connection because it does trust the CA that signed it but the certificate presented to us, does not contain the address (72.163.4.161) that we used to connect out towards the server.



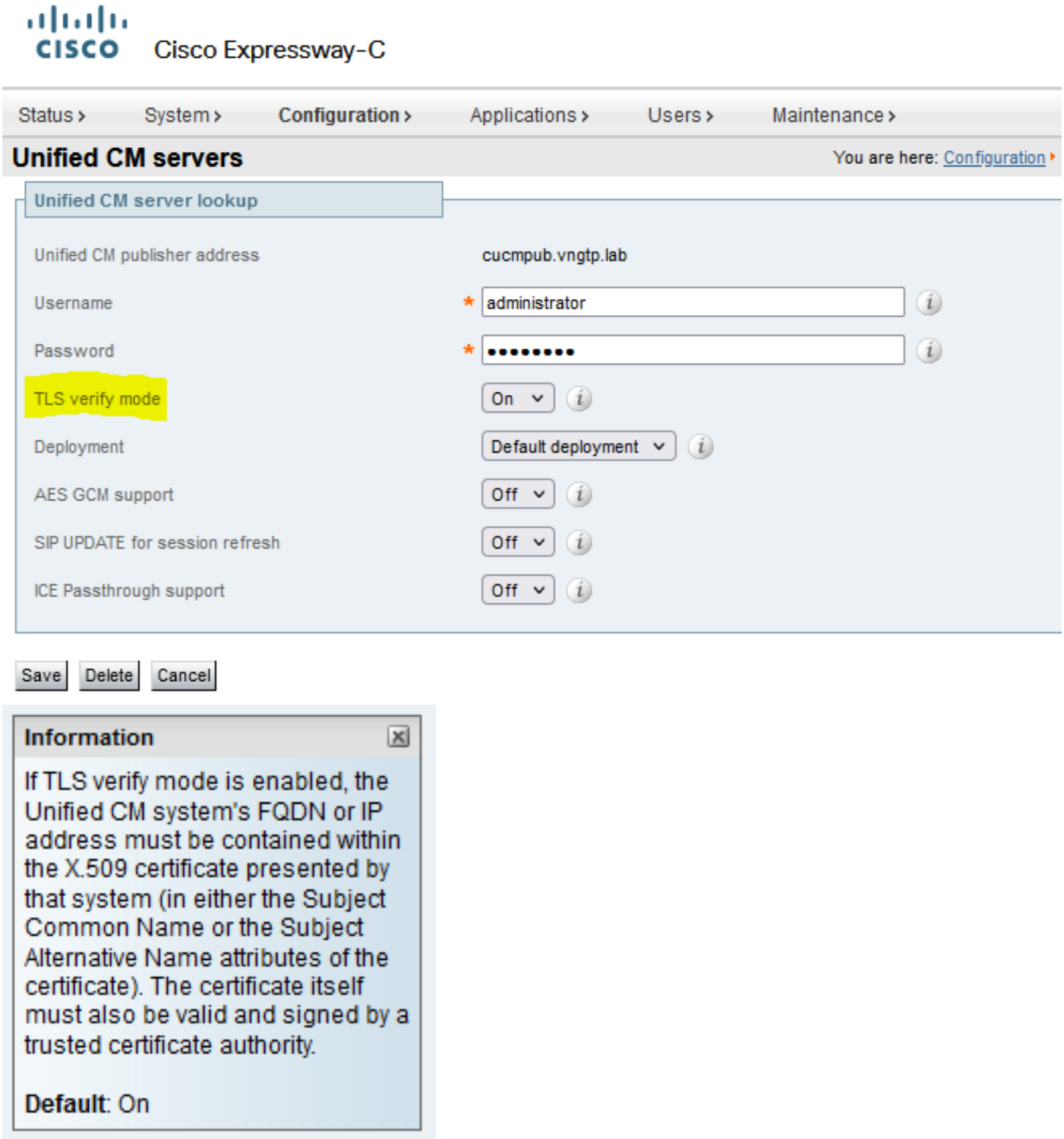
In the browser, you can bypass this however it is a setting that you can enable on TLS connections that a bypass is not allowed. Therefore, it is important that your certificates contain the right CN or SAN names that the remote party plans to use in order to connect to it.

Behavior Change

MRA services rely heavily on several HTTPS connections over the Expressways towards the CUCM / IM&P / Unity servers to authenticate properly and to gather on the right information specific for the client that logs in. This communication usually happens over ports 8443 and 6972.

Versions Lower than X14.2.0

In versions lower than X14.2.0, the traffic server on Expressway-C that handles those secure HTTPS connections did not verify the certificate that was presented by the remote end. This could lead to man-in-the-middle attacks. On the MRA configuration, there is an option for TLS certificate verification by the configuration of the 'TLS Verify Mode' to 'On' when you would add either CUCM / IM&P / Unity servers under **Configuration > Unified Communications > Unified CM servers / IM and Presence Service nodes / Unity Connection servers**. The configuration option and the relevant information box is shown as an example, which indicates that it does verify the FQDN or IP in the SAN as well as the validity of the certificate and whether it is signed by a trusted CA.



This TLS certificate verification check is only done though at the discovery of the CUCM / IM&P / Unity servers and not at the time when during MRA login the various servers are queried. A first drawback of this configuration, is that it only verifies it for the publisher address you add in. It does not validate if the certificate on the subscriber nodes has been correctly set up as it retrieves the subscriber node info (FQDN or IP) from the database of the publisher node. A second drawback of this configuration as well, is that what is advertised over to the MRA clients as the connection information can be different from publisher address that has been put in the Expressway-C configuration. For example on CUCM, under **System > Server** you could advertise the server out with an IP address (10.48.36.215 for example) and this is then used by the MRA clients (via the proxied Expressway connection) however you could add in the CUCM on Expressway-C with the FQDN of cucm.steven.lab. So assume that the tomcat certificate of CUCM contains cucm.steven.lab as SAN entry but not the IP address, then the discovery with 'TLS Verify Mode' set to 'On' succeeds but the actual communications from the MRA clients can target a different

FQDN or IP and thus fail the TLS verification.

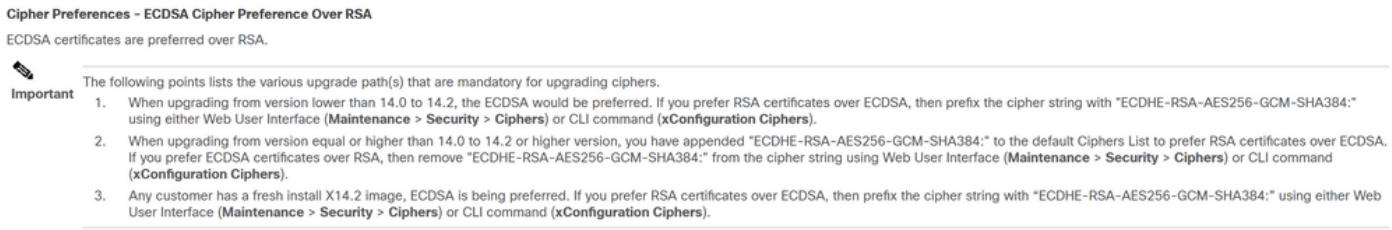
Versions of X14.2.0 and Higher

From X14.2.0 version onwards, the Expressway server does perform on the TLS certificate verification for every single HTTPS request that is made through the traffic server. That means it does also perform this when the 'TLS Verify Mode' is set to 'Off' during the discovery of the CUCM / IM&P / Unity nodes. When the verification does not succeed, the TLS handshake does not complete and the request fails which can lead to loss of functionality like redundancy or failover issues or complete login failures for example. Also with 'TLS Verify Mode' set to 'On', it does not guarantee that all connections do work fine as covered in the example later.

The exact certificates that the Expressway checks towards the CUCM / IM&P / Unity nodes are as shown on the section of the MRA guide.

Aside from the default on TLS verification, there is also a change introduced in X14.2 which could advertise a different preference order for the cipher list, which depends on your upgrade path. This can cause unexpected TLS connections after a software upgrade because it can happen that before the upgrade it requested for the Cisco Tomcat or Cisco CallManager certificate from CUCM (or any other product that has a separate certificate for ECDSA algorithm) but that after the upgrade it requests for the ECDSA variant (which is the more secure cipher variant actually than RSA). The Cisco Tomcat-ECDSA or Cisco CallManager-ECDSA certificates could be signed by a different CA or just still self-signed certificates (the default).

This cipher preference order change is not always relevant to you as it depends on the upgrade path as shown from the Expressway X14.2.1 release notes. In short you can see from **Maintenance > Security > Ciphers** for each of the cipherlists whether it does prepend "ECDHE-RSA-AES256-GCM-SHA384:" or not. If it does not, then it prefers the newer ECDSA cipher over the RSA cipher. If it does, then you have the behavior as previous with RSA that has the higher preference then.



Cipher Preferences - ECDSA Cipher Preference Over RSA

ECDSA certificates are preferred over RSA.

Important

The following points lists the various upgrade path(s) that are mandatory for upgrading ciphers.

1. When upgrading from version lower than 14.0 to 14.2, the ECDSA would be preferred. If you prefer RSA certificates over ECDSA, then prefix the cipher string with "ECDHE-RSA-AES256-GCM-SHA384:" using either Web User Interface (**Maintenance** > **Security** > **Ciphers**) or CLI command (**xConfiguration Ciphers**).
2. When upgrading from version equal or higher than 14.0 to 14.2 or higher version, you have appended "ECDHE-RSA-AES256-GCM-SHA384:" to the default Ciphers List to prefer RSA certificates over ECDSA. If you prefer ECDSA certificates over RSA, then remove "ECDHE-RSA-AES256-GCM-SHA384:" from the cipher string using Web User Interface (**Maintenance** > **Security** > **Ciphers**) or CLI command (**xConfiguration Ciphers**).
3. Any customer has a fresh install X14.2 image, ECDSA is being preferred. If you prefer RSA certificates over ECDSA, then prefix the cipher string with "ECDHE-RSA-AES256-GCM-SHA384:" using either Web User Interface (**Maintenance** > **Security** > **Ciphers**) or CLI command (**xConfiguration Ciphers**).

There are two ways the TLS verification could fail in this scenario, which are covered in detail later:

1. CA that signed the remote certificate is not trusted

a. Self-signed certificate

b. Certificate signed by unknown CA

2. Connection Address (FQDN or IP) is not contained in the certificate

Troubleshoot Scenarios

The next scenarios show up a similar scenario in a lab environment where MRA login did fail after

an upgrade of Expressway from X14.0.7 to X14.2. They share similarities in the logs, however the resolution is different. The logs are just collected by the diagnostic logging (from **Maintenance > Diagnostics > Diagnostic logging**) that got started before the MRA login and stopped after the MRA login failed. No additional debug logging has been enabled for it.

1. CA That Signed The Remote Certificate Is Not Trusted

The remote certificate could either be signed by a CA that is not included in the trust store of the Expressway-C or could be a self-signed certificate (in essence a CA as well) which is not added in the trust store of the Expressway-C server.

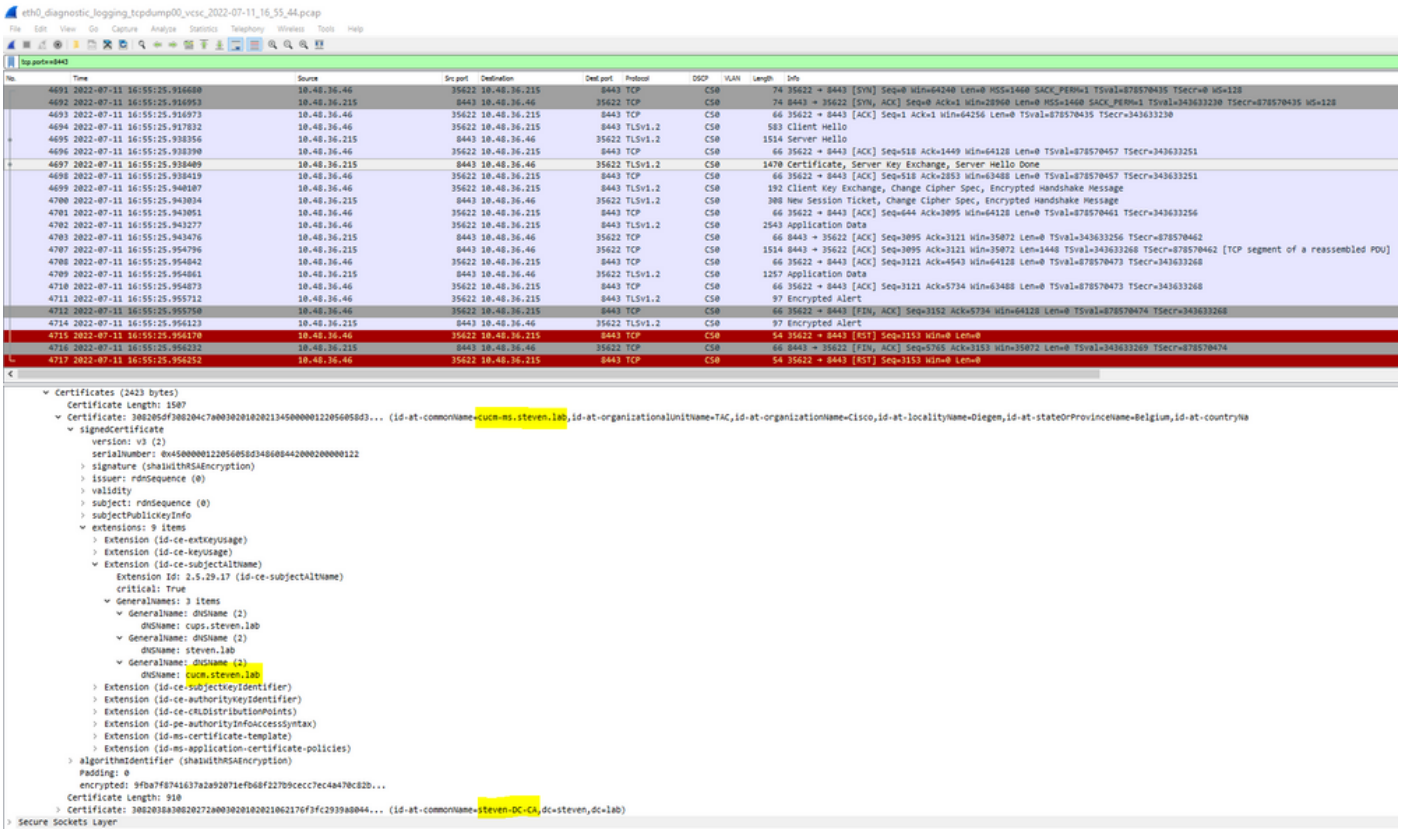
In the example here, you can observe that the requests that go to CUCM (10.48.36.215 - cucm.steven.lab) are handled correctly on port 8443 (200 OK response) but it throws up an error (502 response) on port 6972 for the TFTP connection.

```
===Success connection on 8443===

2022-07-11T18:55:25.910+02:00 vcsc traffic_server[18242]: UTCTime="2022-07-11 16:55:25,910"
Module="network.http.trafficserver" Level="INFO": Detail="Receive Request" Txn-id="189"
TrackingID="6af9a674-9ebc-41ea-868e-90e7309a758c" Src-ip="127.0.0.1" Src-port="35764" Last-via-
addr="" Msg="GET
http://vcs_control.steven.lab:8443/c3RldmVuLmxhYi9odHRwcy9jdWNtLnN0ZXZlbi5sYWIvODQ0Mw/cucm-
uds/user/emusk/devices HTTP/1.1"

2022-07-11T18:55:25.917+02:00 vcsc traffic_server[18242]: Event="Request Allowed" Detail="Access
allowed" Reason="In allow list" Username="emusk" Deployment="1" Method="GET"
Request="https://cucm.steven.lab:8443/cucm-uds/user/emusk/devices"
Rule="https://cucm.steven.lab:8443/cucm-uds/user/" Match="prefix" Type="Automatically generated
rule for CUCM server" UTCTime="2022-07-11 16:55:25,916"
2022-07-11T18:55:25.917+02:00 vcsc traffic_server[18242]: UTCTime="2022-07-11 16:55:25,916"
Module="network.http.trafficserver" Level="INFO": Detail="Sending Request" Txn-id="189"
TrackingID="6af9a674-9ebc-41ea-868e-90e7309a758c" Dst-ip="10.48.36.215" Dst-port="8443" Msg="GET
/cucm-uds/user/emusk/devices HTTP/1.1"
2022-07-11T18:55:25.955+02:00 vcsc traffic_server[18242]: UTCTime="2022-07-11 16:55:25,955"
Module="network.http.trafficserver" Level="INFO": Detail="Receive Response" Txn-id="189"
TrackingID="" Src-ip="10.48.36.215" Src-port="8443" Msg="HTTP/1.1 200 "
2022-07-11T18:55:25.956+02:00 vcsc traffic_server[18242]: UTCTime="2022-07-11 16:55:25,955"
Module="network.http.trafficserver" Level="INFO": Detail="Sending Response" Txn-id="189"
TrackingID="" Dst-ip="127.0.0.1" Dst-port="35764" Msg="HTTP/1.1 200 "

===Failed connection on 6972===

2022-07-11T18:55:26.000+02:00 vcsc traffic_server[18242]: UTCTime="2022-07-11 16:55:26,000"
Module="network.http.trafficserver" Level="INFO": Detail="Receive Request" Txn-id="191"
TrackingID="bb0c8492-8c15-4537-a7d1-082dde781dbd" Src-ip="127.0.0.1" Src-port="35766" Last-via-
addr="" Msg="GET
http://vcs_control.steven.lab:8443/c3RldmVuLmxhYi9odHRwcy9jdWNtLnN0ZXZlbi5sYWIvNjk3Mg/CSFemusk.c
nf.xml HTTP/1.1"

2022-07-11T18:55:26.006+02:00 vcsc traffic_server[18242]: UTCTime="2022-07-11 16:55:26,006"
Module="network.http.trafficserver" Level="INFO": Detail="Sending Request" Txn-id="191"
TrackingID="bb0c8492-8c15-4537-a7d1-082dde781dbd" Dst-ip="10.48.36.215" Dst-port="6972" Msg="GET
/CSFemusk.cnf.xml HTTP/1.1"
2022-07-11T18:55:26.016+02:00 vcsc traffic_server[18242]: UTCTime="2022-07-11 16:55:26,016"
Module="network.http.trafficserver" Level="INFO": Detail="Sending Request" Txn-id="191"
TrackingID="bb0c8492-8c15-4537-a7d1-082dde781dbd" Dst-ip="10.48.36.215" Dst-port="6972" Msg="GET
/CSFemusk.cnf.xml HTTP/1.1"
2022-07-11T18:55:26.016+02:00 vcsc traffic_server[18242]: [ET_NET 0] WARNING: Core server
certificate verification failed for (cucm.steven.lab). Action=Terminate Error=self signed
```

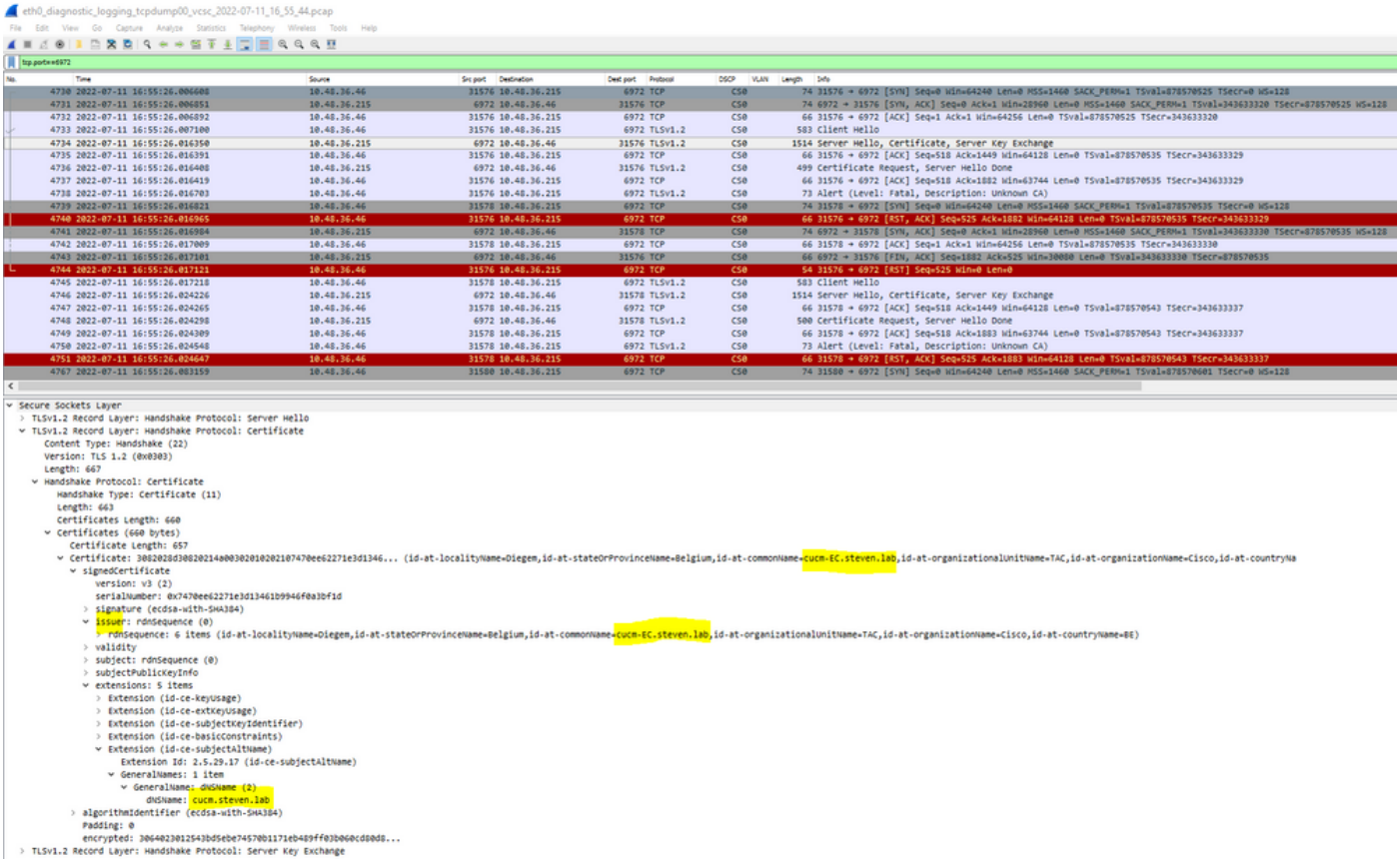
certificate server=cucm.steven.lab(10.48.36.215) depth=0
2022-07-11T18:55:26.016+02:00 vcsc traffic_server[18242]: [ET_NET 0] **ERROR: SSL connection failed for** 'cucm.steven.lab': error:1416F086:**SSL routines:tls_process_server_certificate:certificate verify failed**
2022-07-11T18:55:26.024+02:00 vcsc traffic_server[18242]: UTCTime="2022-07-11 16:55:26,024" Module="network.http.trafficserver" Level="INFO": Detail="Sending Response" Txn-id="191" TrackingID="" Dst-ip="127.0.0.1" Dst-port="35766" Msg="HTTP/1.1 **502 connect failed**"

The error of 'certificate verify failed' indicates the fact that the Expressway-C could not validate the TLS handshake. The reason for it, is shown on the warning line as it indicates a self signed certificate. If the depth is shown as 0, it is a self signed certificate. When the depth is higher than 0, it means that it has a certificate chain and thus it is signed by an unknown CA (from the perspective of Expressway-C).

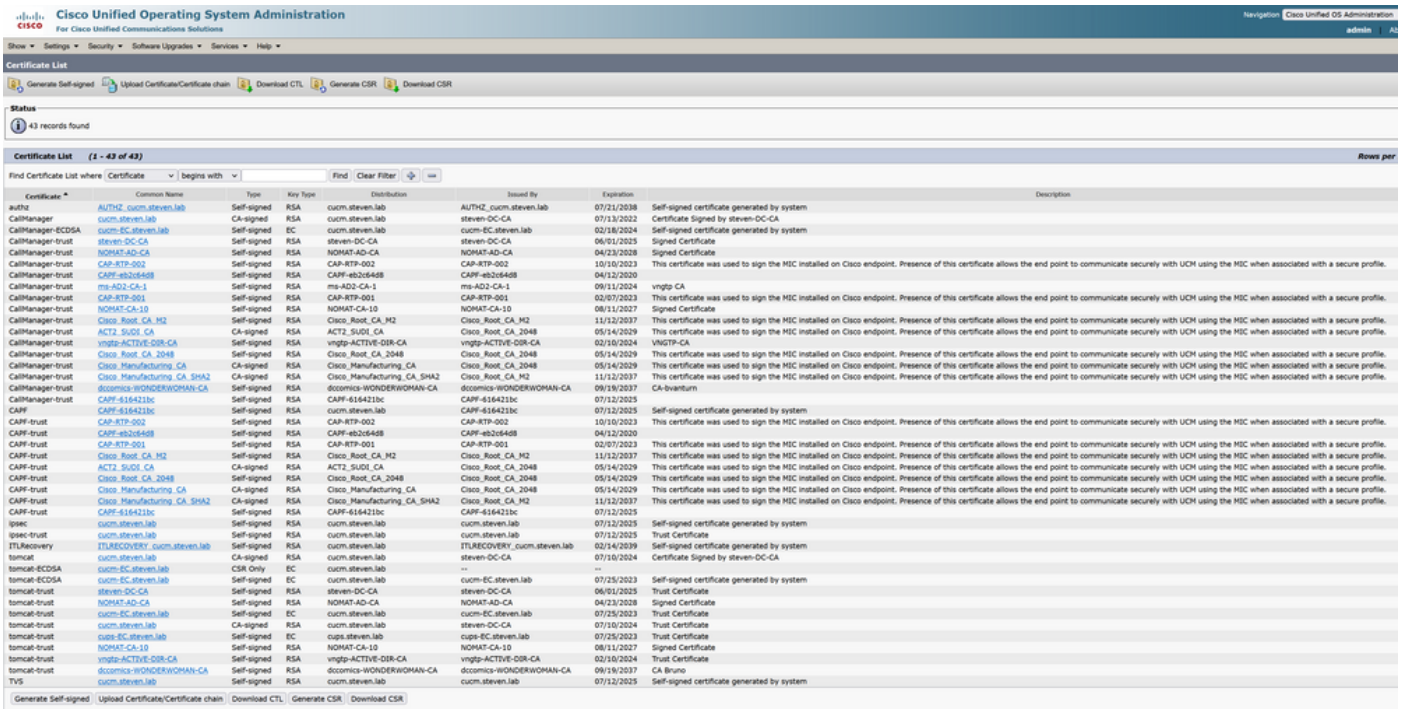
When we look in the pcap file that got collected at the timestamps mentioned from the text logs, you can see that CUCM presents the certificate with CN as cucm-ms.steven.lab (and cucm.steven.lab as SAN) signed by steven-DC-CA to the Expressway-C on port 8443.



But when we inspect the certificate presented on port 6972, you can see it is a self-signed certificate (Issuer is itself) with CN set up as cucm-EC.steven.lab. The -EC extension gives the indication that this is the ECDSA certificate set up on CUCM.



On CUCM under Cisco Unified OS Administration, you can look at the certificates in place under Security > Certificate Management as shown for example here. It shows up a different certificate for tomcat and tomcat-ECDSA where the tomcat is CA signed (and trusted by the Expressway-C) while the tomcat-ECDSA certificate is self-signed and not trusted by the Expressway-C here.



2. Connection Address (FQDN Or IP) Is Not Contained in The Certificate

Aside from the trust store, there traffic server also verifies the connection address that the MRA client makes the request towards. For example, when you have set up on CUCM under **System > Server** your CUCM with the IP address (10.48.36.215), then the Expressway-C advertises this as

such to the client and subsequent requests from the client (proxied through the Expressway-C) are targetted towards this address.

When that particular connection address is not contained within the server certificate, the TLS verification fails as well and a 502 error is thrown that results in MRA login failure for example.

```
2022-07-11T19:49:01.472+02:00 vcsc traffic_server[3916]: UTCTime="2022-07-11 17:49:01,472"
Module="network.http.trafficserver" Level="DEBUG": Detail="Receive Request" Txn-id="144"
TrackingID="0a334fa8-41e9-4b97-adf4-e165372c38cb" Src-ip="127.0.0.1" Src-port="30044" Last-via-
addr=""
HTTPMSG:
|GET http://vcs_control.steven.lab:8443/c3RldmVuLmxhYi9odHRwcy8xMC40OC4zNi4yMTUvODQ0Mw/cucm-
uds/user/emusk/devices?max=100 HTTP/1.1
...

2022-07-11T19:49:01.478+02:00 vcsc traffic_server[3916]: UTCTime="2022-07-11 17:49:01,478"
Module="network.http.trafficserver" Level="INFO": Detail="Sending Request" Txn-id="144"
TrackingID="0a334fa8-41e9-4b97-adf4-e165372c38cb" Dst-ip="10.48.36.215" Dst-port="8443" Msg="GET
/cucm-uds/user/emusk/devices?max=100 HTTP/1.1"
2022-07-11T19:49:01.478+02:00 vcsc traffic_server[3916]: UTCTime="2022-07-11 17:49:01,478"
Module="network.http.trafficserver" Level="DEBUG": Detail="Sending Request" Txn-id="144"
TrackingID="0a334fa8-41e9-4b97-adf4-e165372c38cb" Dst-ip="10.48.36.215" Dst-port="8443"
HTTPMSG:
|GET /cucm-uds/user/emusk/devices?max=100 HTTP/1.1
...

2022-07-11T19:49:01.491+02:00 vcsc traffic_server[3916]: [ET_NET 2] WARNING: SNI (10.48.36.215)
not in certificate. Action=Terminate server=10.48.36.215(10.48.36.215)
2022-07-11T19:49:01.491+02:00 vcsc traffic_server[3916]: [ET_NET 2] ERROR: SSL connection failed
for '10.48.36.215': error:1416F086:SSL routines:tls_process_server_certificate:certificate
verify failed
```

Where c3RldmVuLmxhYi9odHRwcy8xMC40OC4zNi4yMTUvODQ0Mw translates (base64 - https://www.base64decode.org/) to steven.lab/https/10.48.36.215/8443, which shows that it must make the connection towards 10.48.36.215 as the connection address rather than to cucm.steven.lab. As shown in the packet captures, the CUCM tomcat certificate does not contain the IP address in the SAN and thus the error is thrown.

How to Validate It Easily

You can validate whether you run into this behavior change easily with the next steps:

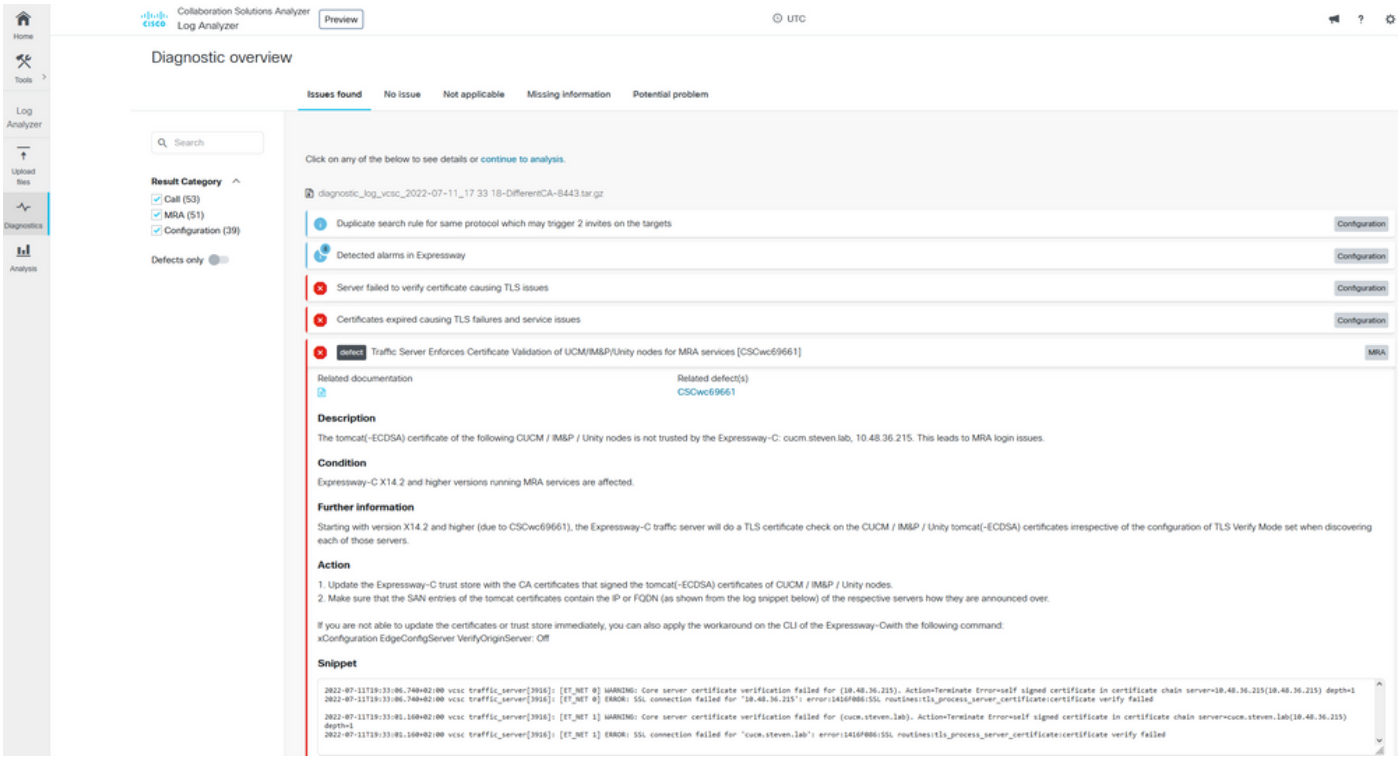
1. Start diagnostic logging on Expressway-E and C server(s) (ideally with TCPDumps enabled) from **Maintenance > Diagnostics > Diagnostic Logging** (in case of a cluster, it is sufficient to start it from the primary node)

2. Attempt a MRA login or test the broken functionality after the upgrade

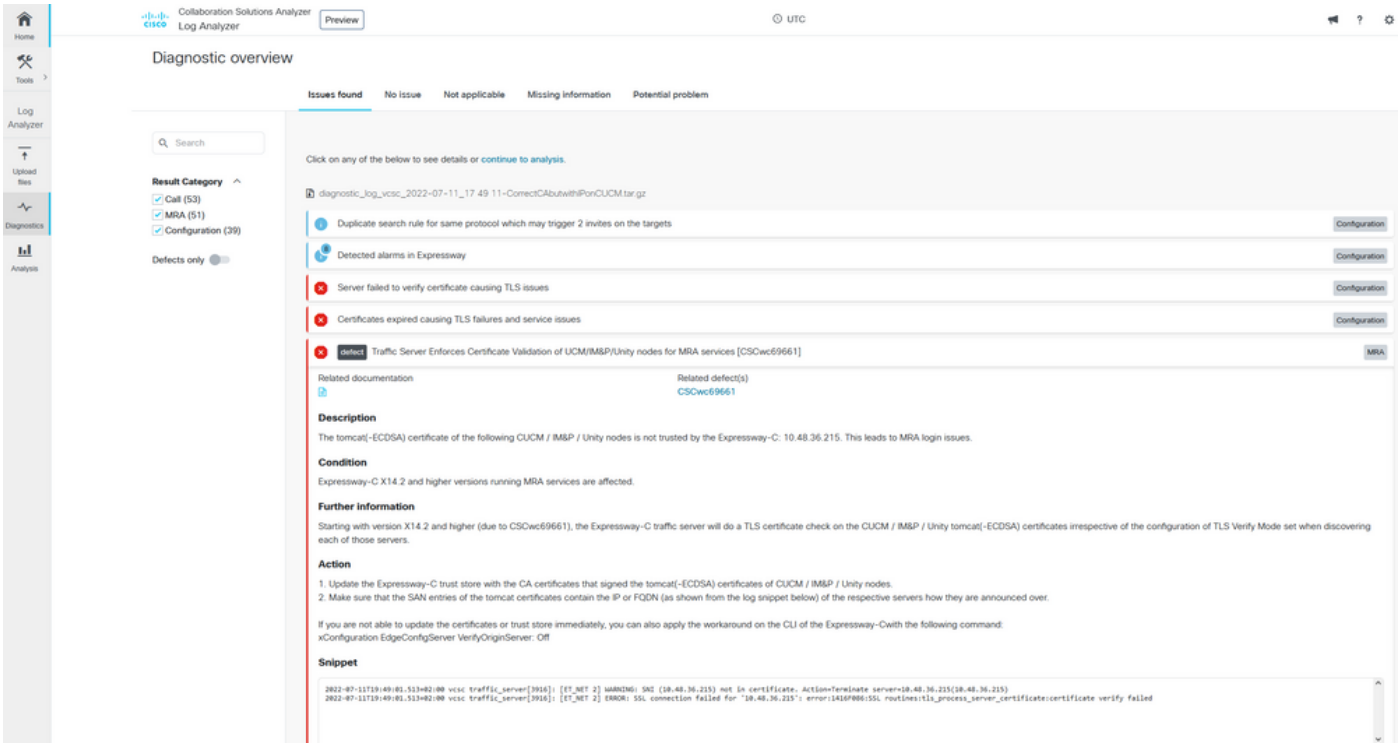
3. Wait until it fails and then stop the diagnostic logging on Expressway-E and C server(s) (in case of a cluster, make sure to collect the logs from every single node of the cluster individually)

4. Upload and analyze the logs on the Collaboration Solution Analyzer tool

5. If you run into the issue, it picks up the most recent warning and error lines that relate to this change for each of the servers affected



CA diagnostic signature



SNI Diagnostic Signature

Solution

The long term solution is to make sure that the TLS verification works out fine. What action to perform depends on the warning message displayed.

When you observe the *WARNING: Core server certificate verification failed for (<server-FQDN-or-IP>). Action=Terminate Error=self signed certificate server=cucm.steven.lab(10.48.36.215)* ***depth=x*** message, then you need to update the trust store on the Expressway-C servers accordingly. Either with the CA chain that signed this certificate (depth > 0) or with the self-signed certificate (depth = 0) from **Maintenance > Security > Trusted CA Certificate**. Make sure to

perform on this action on every server in the cluster. Another option would be to sign the remote certificate by a known CA on the Expressway-C trust store.

> **Note: Expressway does not allow to upload two different (self-signed for example) certificates into the trust store of Expressway that have the same Common Name (CN) as per Cisco bug ID CSCwa12905. In order to correct on this, move to CA-signed certificates or upgrade your CUCM to version 14 where you can re-use the same (self-signed) certificate for Tomcat and CallManager.**

When you observe the *WARNING: SNI (<server-FQDN-or-IP>) not in certificate* message, then it indicates that this server FQDN or IP is not contained within the certificate that got presented. Either you can adapt the certificate to include that information or you can modify the configuration (like on CUCM on System > Server to something that is contained in the server certificate) and then refresh the configuration on the Expressway-C server for it to be taken into account.

The short term solution is to apply the workaround as documented to fallback to the previous behavior before X14.2.0. You can perform on this through the CLI on the Expressway-C server nodes with the newly introduced command:

```
xConfiguration EdgeConfigServer VerifyOriginServer: Off
```