Troubleshoot Expressway Traffic Server Certificate Verification for MRA Services

Contents

Introduction
<u>Prerequisites</u>
Requirements
Components Used
Background Information
Trusted CA Chain
SAN or CN Check
Behavior Change
Versions Lower than X14.2.0
Versions of X14.2.0 and Higher
Troubleshoot Scenarios
1. CA that Signed the Remote Certificate is not Trusted
2. Connection Address (FQDN Or IP) is not Contained in the Certificate
How to Validate It Easily
<u>Solution</u>
Related Information

Introduction

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

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

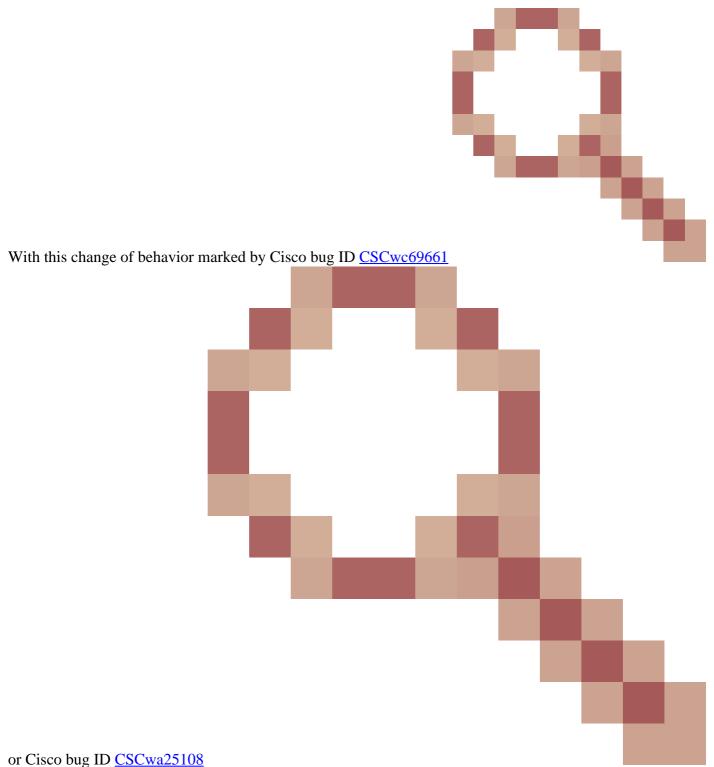
- Expressway basic configuration
- MRA basic configuration

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



, 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. This servers two purposes: authentication (to know who the remote party is you connect to) and privacy (the encryption). The authentication protects against man-inthe-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

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, 0=Cisco, 0U=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, 0=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 <u>CSCwa12905</u> 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.

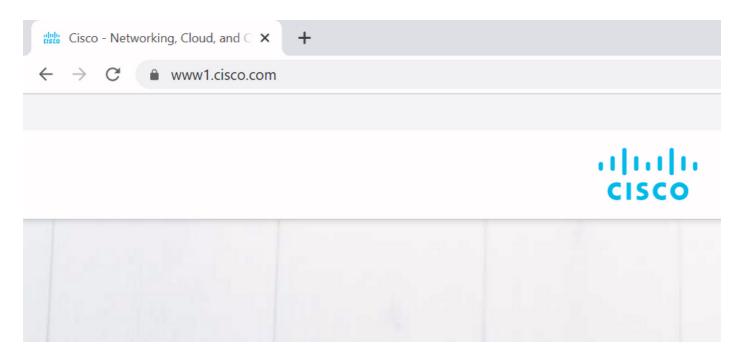
The same kind of operation happens in your browser so let us look into this through an example. If you browse to <u>https://www.cisco.com</u> you see a lock icon next to the URL you entered and it means that it is a trusted connection. This is based both on the CA trust chain (from first section) as 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 <u>www.cisco.com</u> 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).

ertificate	×		
neral Details Certification Path		cisco	Products and Servi
Certificate Information			
This certificate is intended for the follow • Ensures the identity of a remote compu • Proves your identity to a remote compu • 2.16.840.1.113839.0.6.3 • 2.23.140.1.2.2	ter		
* Refer to the certification authority's statement	nt for details.		
Issued to: www.cisco.com			
Issued by: HydrantID Server CA O1			
Valid from 2/16/2022 to 2/16/2023			

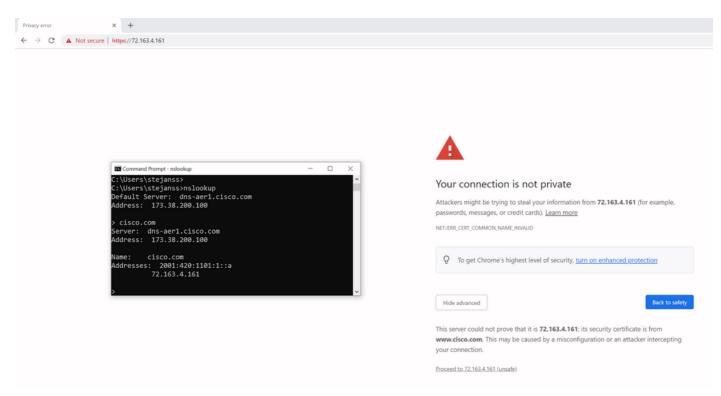
When we look to the details of the certificate and in particular to the SAN entries, we see that the same is repeated as well as some other FQDNs:

	ation Path	
how: <all></all>	\sim	
CRL Distribution P	Value [1]Certificate Policy: [1]CRL Distribution DNS Name=cisco-i	^
Subject Key Identi Enhanced Key Usag SCT List	 b18ceccd49a5dfd74 Server Authenticatio v1, adf7befa7cff10c 	
Key Usage Thumbprint	Digital Signature, Ke Odddb6ce30b00bd7	~
DNS Name=cisco-imag DNS Name=cisco.com DNS Name=www-01.c DNS Name=www-02.c DNS Name=www-rtp.c DNS Name=www.cisco DNS Name=www.cisco DNS Name=www.stati DNS Name=www1.cisc	isco.com isco.com isco.com o.com lafiles-cisco.com c-cisco.com	
DNS Name=www1.054	Edit Properties	Copy to File

This means that when we would request to connect to <u>https://www1.cisco.com</u> 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 <u>https://www.cisco.com</u> but rather directly to the IP address (<u>https://72.163.4.161</u>), 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.

CISCO Cisco Expressway-C

Status >	System >	Configuration >	Applications >	Users >	Maintenance >
Unified (CM servers				You are here: Configuration
Unified C	M server lookup)			
Unified CM	publisher address	S	cucmpub.vngtp.lai	b	
Username			* administrator		i
Password			*		(i)
TLS verify	mode		On v (i)		
Deploymer	nt		Default deployme	ent v i	
AES GCM	support		Off v (i)		
SIP UPDAT	E for session refr	esh	Off v (i)		
ICE Passth	rough support		Off v i		



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 <u>MRA guide</u>.

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 <u>release notes</u>. 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 Preference Over RSA

 ECDSA certificates are preferred over RSA.

 ECDSA certificates are preferred over RSA.

 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, then prefix the cipher string with "ECDHE-RSA-AES256-GCM-SHA384:" to prefer RSA certificates over ECDSA.

 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.

<#root>

===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="net

```
2022-07-11T18:55:25.917+02:00 vcsc traffic_server[18242]: Event="Request Allowed" Detail="Access allow
2022-07-11T18:55:25.917+02:00 vcsc traffic_server[18242]: UTCTime="2022-07-11 16:55:25,916" Module="netw
2022-07-11T18:55:25.955+02:00 vcsc traffic_server[18242]: UTCTime="2022-07-11 16:55:25,955" Module="netw
2022-07-11T18:55:25.956+02:00 vcsc traffic_server[18242]: UTCTime="2022-07-11 16:55:25,955" Module="netw
```

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="net

2022-07-11T18:55:26.006+02:00 vcsc traffic_server[18242]: UTCTime="2022-07-11 16:55:26,006" Module="net 2022-07-11T18:55:26.016+02:00 vcsc traffic_server[18242]: UTCTime="2022-07-11 16:55:26,016" Module="net 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="net 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.

1 🐵 🔰 🗅 🗙 🛤 Q 👄 🖶 🖼 ∓ 🚊 🛄					
	🔲 @ @ @ II				
tx+8443					
Time	Source	Src port Destination	Dest port Protocol	DSCP VLAN	Length Info
4691 2022-07-11 16:55:25.916680	10.48.36.46	35622 10.48.36.215	8443 TCP	C50	74 35622 + 8443 [SYN] Seque Win+64240 Len+0 MSS=1460 SACK PERM=1 TSval=878570435 TSecr+0 MS=128
4692 2022-07-11 16:55:25.916953	10.48.36.215	8443 10.48.36.46	35622 TCP	CSB	74 8443 + 35622 [SYN, ACK] Seq+0 Ack+1 Win+28960 Len+0 MSS=1460 SACK_PERM+1 T5val+343633230 TSecr+878570435 WS+128
4693 2022-07-11 16:55:25.916973	10.48.36.46	35622 10.48.36.215	8443 TCP	C58	66 35622 → 8443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=878570435 TSecr=343633230
4694 2022-07-11 16:55:25.917832	10.48.36.46	35622 10.48.36.215	8443 TLSv1.2	CS0	S83 Client Hello
4695 2022-07-11 16:55:25.938356	10.48.36.215	8443 10.48.36.46	35622 TLSv1.2	CS0	1514 Server Hello
4696 2022-07-11 16:55:25.938390	10.48.36.46	35622 10.48.36.215	8443 TCP	CS0	66 35622 → 8443 [ACK] Seq=518 Ack=1449 Win=64128 Len=0 TSval=878570457 TSecr=343633251
4697 2022-07-11 16:55:25.938409	10.48.36.215	8443 10.48.36.46	35622 TLSv1.2	CSB	1470 Certificate, Server Key Exchange, Server Hello Done
4698 2022-07-11 16:55:25.938419	10.48.36.46	35622 10.48.36.215	8443 TCP	CS0	66 35622 + 8443 [ACK] Seq=518 Ack=2853 Win=63488 Len=0 TSval=878570457 TSecr=343633251
4699 2022-07-11 16:55:25.940107	10.48.36.46	35622 10.48.36.215	8443 TLSv1.2	CS0	192 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
4700 2022-07-11 16:55:25.943034	10.48.36.215	8443 10.48.36.46	35622 TLSv1.2	CSB	308 New Session Ticket, Change Cipher Spec, Encrypted Handshake Hessage
4701 2022-07-11 16:55:25.943051	10.48.36.46	35622 10.48.36.215	8443 TCP	CS0	66 35622 → 8443 [ACK] Seq=644 Ack=3095 Win=64128 Len=0 TSVal=878578461 TSecr=343633256
4702 2022-07-11 16:55:25.943277	10.48.36.46	35622 10.48.36.215	8443 TLSv1.2	CS0	2543 Application Data
4703 2022-07-11 16:55:25.943476	10.48.36.215	8443 10.48.36.46	35622 TCP	CS0	66 8443 + 35622 [ACK] Seq=3095 Ack=3121 Win=35072 Len=0 TSval=343633256 TSecr=878570462
4707 2022-07-11 16:55:25.954796	10.48.36.215	8443 10.48.36.46	35622 TCP	CSB	1514 8443 + 35622 [ACK] Seq=3095 Ack=3121 Win=35072 Len=1448 TSval=343633268 TSecr=878570462 [TCP segment of a reassembled
4708 2022-07-11 16:55:25.954842	10.48.36.46	35622 10.48.36.215	8443 TCP	CS0	66 35622 + 8443 [ACK] Seq+3121 Ack+4543 Win+64128 Len+0 TSval=878570473 TSecr=343633268
4709 2022-07-11 16:55:25.954861	10.48.36.215	8443 10.48.36.46	35622 TL5v1.2	C50	1257 Application Data
4710 2022-07-11 16:55:25.954873	10.48.36.46	35622 10.48.36.215	8443 TCP	CSB	66 35622 → 8443 [ACK] Seq=3121 Ack+5734 Win=63488 Len+0 TSval=878570473 TSecr=343633268
4711 2022-07-11 16:55:25.955712	10.48.36.46	35622 10.48.36.215	8443 TLSv1.2	CS0	97 Encrypted Alert
4712 2022-07-11 16:55:25.955750	10.48.36.46	35622 10.48.36.215	8443 TCP	CSB	66 35622 + 8443 [FIN, ACK] Seq=3152 Ack=5734 win=64128 Len=0 TSval=878570474 TSecr=343633268
4714 2022-07-11 16:55:25.956123	10.48.36.215	8443 10.48.36.46	35622 TL5v1.2	CSB	97 Encrypted Alert
4715 2022-07-11 16:55:25.956170	10.48.36.46 10.48.36.215	35622 10.48.36.215	8443 TCP 35622 TCP	CS8 CS8	54 35622 + 8443 [#ST] Seq#3153 Win+0 Len+0 66 8443 + 35622 [FIN, ACK] Seq#5765 Ack#3153 Win#35072 Len+0 TSval#343633269 TSecr#878570474
4716 2022-07-11 16:55:25.956232 4717 2022-07-11 16:55:25.956252	10.48.36.46	8443 10.48.36.46 35622 10.48.36.215	8443 TCP	CSB	66 6443 4 35022 (FLM, ACK) SCH05/05 ACK45153 MINH55072 LENNE 15/015345633209 1502748/05/04/4 54 35622 + 8443 [STI] Scy-3153 Winhe Lenne
 signedCertificate version: v3 (2) 		d-at-commonName <mark>«cucm-ms.steven.la</mark>	<mark>b</mark> ,id-at-organizationalu	nitWame=TAC,id	-at-organizationwame-Cisco,id-at-localityWame-Diegen,id-at-stateO-ProvinceWame-Belgium,id-at-countryWa
<pre>Certificate Length: 1507 Certificate: 100205/002046700010201 vignedcertificate version: v3 (2) serialnuber: 0x4500000122056653 signature (1x4514/ht534ncryption issue: rdd56quence (0) validity subject: rdd5equence (0)</pre>	d348668442000200000122	d-at-commonName <mark>-cucm-ms.steven.la</mark>	<mark>e</mark> ,id-at-organizationalu	nitwame=TAC,id	-at-organizationwame-Cisco,id-at-localitywame-Djegem,id-at-stateOrProvincewame-Belgium,id-at-countryWa
Certificate Length: 1597 v Certificate: 308206/780820c78093001 v signedcertificate version: v3 (2) serialNumber: dosSome012205605) signeture (shalidthSkiEncryston) subject: rdnSequence (0)) subject: rdnSequence (0)	4348668442608268686122)) me)	d-at-componiane- <mark>duce-ind, ite-en, la</mark>	8,id-at-organizationalu	nitWame∘TAC,id	-at-organizationwame-Cisco,id-at-localitywame-Diegen,id-at-stateOrProvincewame-Belgium,id-at-countrywa
Certificate Length: 159 v Certificate: 100206/100204/2003003 v signedCertificate version:v0 (2) serialNumber: 6x4500001206566) signedCertificate validity) subjectivalIcevers() validity) subjectivalIcevers() i stension (16-ce-extrayusage)) stension (16-ce-extrayusage) v tension (16-ce-extrayusage)	d346665442000200000122)) ne) -ce-subjectAltName)	d-at-commonName <mark>-suce-ms,ite-en,la</mark>	8,id-ət-organizətionəlu	nitwame-TAC,id	-at-organizationwame-Cisco,id-at-localitywame-Diegen,id-at-stateOrProvincewame-Belgium,id-at-countrywa
Certificate Length: 1597 v Certificate: 302036/100204/20030201 v signedcertificate version: v0 (2) serialNumber: doddemostl2006065) signet: doddemoste) signet: doddemoste) subjectholice(pin6 version: di-ce-extsyusge)) Ditension (di-ce-extsyusge) v Extension (di-ce-extsyusge) v Extension (di-ce-subjectAlbum Extension (di-ce-	d3466644200020000122) ne) -ce-subjectAltName) ab	6-at-componiane- <mark>suce-ins, ite-en, la</mark>	8,id-ət-organizətionəlu	nitwame-TAC,id	-at-organizationwame-Cisco,id-at-localitywame-Djegem,id-at-stateOrProvincewame-Belgium,id-at-countrywa
Certificate iength: 1597 v Certificate: 100206/T00204/20003000 v signedCertificate version:v0 (2) serialNumber: 6x45600012265665) signet: 6x56000012666848) ulister: 6x560000000) ulister: 6x5600000000) ulisterNollidersinfo version:s: 9 items) totension (16-ce-setsyusage) v totension (16-ce-subjectifue totension (16-ce-subjectifue version: 1 items version: 1 items versi	d34666442000200000122) ne) e-subjectAltName) ab ab motifier) conditier) conditier)	s-at-commoniane <mark>-cuca-as (teven-la</mark>	8,id-ət-organizətionəlu	mitwame=TAC,id	-at-organizationwame-Cisco,id-at-localitywame-Diegen,id-at-stateOrProvinceWame-Belgium,id-at-countrywa
Certificate Length: 1597 v Certificate: 100206/T00206/T000206/T000206/T signedCertificate version:v0 (2) serialNumber: dwsG000012205665) signetwr: chasidtheSkerrytion) subject: rdnsquence (0)) subject: rdnsquence (0) v Extension (16-ce-kryunage) v GeneralNume: Steven.lbb v GeneralNume: Steven.lbb v GeneralNume: Steven.lbb v GeneralNume: Steven.lbb v GeneralNume: Steven.lbb v GeneralNume: Steven.lbb v GeneralNume: Steven.lbb	d34666544200020000122) me) ce-subjectAltwame) ab ab ab ab ab ab ab ab ab ab ab ab ab	d-at-comoniane <mark>-suca-as,iteven,la</mark>	8,id-ət-organizətionəlu	nitvame-TAC,id	∙at-organizationuame-Cisco,id-at-localityName-Diegen,id-at-stateOrProvinceName-Belgium,id-at-countryNa

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.

∠ ● ■ 2 2 2 9 + + 5 7 ± 🚍	= 4441				
porte=6972					
4730 2022-07-11 16:55:26.006608	10.48.36.46	Sec port Cestination 31576 10.48.36.215	Cest port Protocol 6972 TCP	CS0 VUN	Length 5de 74 31576 ★ 6972 [SYN] Seque Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSV8]=878578525 TSecn=0 MS=128
4731 2022-07-11 16:55:26.006851	10.48.36.215	6972 10.48.36.46	31576 TCP	CSB	74 6972 + 31576 [SVN, ACK] Seq=0 ACk=1 kin=28960 Len=0 MSS=1460 SACK PERM=1 TSV81=343633320 TSecr=878570525 W
4732 2022-07-11 16:55:26.006892	10.48.36.46	31576 10.48.36.215	6972 TCP	CS0	66 31576 → 6972 [ACK] Seg=1 Ack=1 Win=64256 Len=0 TSval=878570525 TSecr=343633320
4733 2022-07-11 16:55:26.007100	10.48.36.46	31576 10.48.36.215	6972 TLSv1.2	CS0	583 Client Hello
4734 2022-07-11 16:55:26.016350	10.48.36.215	6972 10.48.36.46	31576 TLSv1.2	CS0	1514 Server Hello, Certificate, Server Key Exchange
4735 2022-07-11 16:55:26.016391	10.48.36.46	31576 10.48.36.215	6972 TCP	CS8	66 31576 + 6972 [ACK] Seq=518 Ack+1449 Win+64128 Len+0 TSval=878570535 TSecr=343633329
4736 2022-07-11 16:55:26.016408	10.48.36.215	6972 10.48.36.46	31576 TLSv1.2	CSB	499 Certificate Request, Server Hello Done
4737 2022-07-11 16:55:26.016419	10.48.36.46	31576 10.48.36.215	6972 TCP	C58	66 31576 + 6972 [ACK] Seq=518 Ack=1882 Win=63744 Len=0 TSval=878578535 TSecr=343633329
4738 2022-07-11 16:55:26.016703	10.48.36.46	31576 10.48.36.215	6972 TL5v1.2	cse	73 Alert (Level: Fatal, Description: Unknown CA)
4739 2022-07-11 16:55:26.016821	10.48.36.46	31578 10.48.36.215	6972 TCP	CS0	74 31578 → 6972 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSV8]=878570535 TSecr=0 MS=128
4740 2022-07-11 16:55:26.016965	10.48.36.46	31576 10.48.36.215	6972 TCP 31578 TCP	CS0 CS0	66 31576 + 6972 [RST, ACK] Seq+525 Ack+1882 Win+64128 Len+0 TSval+878570535 TSecr+343633329
4741 2022-07-11 16:55:26.016984 4742 2022-07-11 16:55:26.017009	10.48.36.215 10.48.36.46	6972 10.48.36.46 31578 10.48.36.215	6972 TCP	C50	74 6972 + 31578 [SYN, ACK] Seq+0 Ack+1 Win+28960 Len+0 MSS+1460 SACK_PERN+1 TSval+343633330 TSecr+878570535 W 66 31578 → 6972 [ACK] Seq+1 Ack+1 Win+64256 Len+0 TSval+878570535 TSecr+343633330
4743 2022-07-11 16:55:26.017007	10.48.36.215	6972 10.48.36.46	31576 TCP	CSB	66 6972 + 31576 [FIN, ACK] Seq#1882 ACk#525 Win#300808 Len#0 TSV81#343633330 TSecr#878570535
4744 2022-07-11 16:55:26.017121	10,48,36,46	31576 10,48,36,215	6972 TCP	650	54 31576 + 6972 [RST] SequS25 Wink0 Lenk0
4745 2022-07-11 16:55:26.017218	10.48.36.46	31578 10.48.36.215	6972 TLSV1.2	CS8	583 Client Hello
4746 2022-07-11 16:55:26.024226	10.48.36.215	6972 10.48.36.46	31578 TLSV1.2	CS0	1514 Server Hello, Certificate, Server Key Exchange
4747 2022-07-11 16:55:26.024265	10.48.36.46	31578 10.48.36.215	6972 TCP	CS0	66 31578 → 6972 [ACK] Seq=518 ACk=1449 kin=64128 Len=0 TSval=878570543 TSecr=343633337
4748 2022-07-11 16:55:26.024298	10.48.36.215	6972 10.48.36.46	31578 TLSv1.2	CS0	500 Certificate Request, Server Hello Done
4749 2022-07-11 16:55:26.024309	10.48.36.46	31578 10.48.36.215	6972 TCP	CS0	66 31578 → 6972 [ACK] Seq=518 Ack+1883 Win+63744 Len+0 TSval=878570543 TSecr=343633337
4750 2022-07-11 16:55:26.024548	10.48.36.46	31578 10.48.36.215	6972 TLSv1.2	CS0	73 Alert (Level: Fatal, Description: Unknown CA)
4751 2022-07-11 16:55:26.024647	10.48.36.46	31578 10.48.36.215	6972 TCP	CS8	66 31578 → 6972 [RST, ACK] Seq=525 Ack=1883 Win=64128 Len=0 TSval=878578543 TSecr=343633337
4767 2022-07-11 16:55:26.083159	10.48.36.46	31580 10.48.36.215	6972 TCP	CSB	74 31580 + 6972 [SYN] Seq#0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=878570601 TSecr=0 MS=128
ure Sockets Layer TLSV1.2 Record Layer: Handshake Protocol: Se TLSV1.2 Record Layer: Handshake Protocol: Ce Content Type: Handshake (22) Version: TLS 1.2 (200303) Length: 667					
TLSV1.2 Record Layer: Handshake Protocol: Se TLSV1.2 Record Layer: Handshake Protocol: Ce Content Type: Handshake (22) Version: TLS 1.2 (200303) Length: 667 + Mandshake Protocol: Certificate Handshake Type: Certificate (11)					
LSV1.2 Record Leyer: Handshake Protocol: Se LSV1.2 Record Leyer: Handshake Protocol: Ce Content Type: Handshake (22) Version: TLS 1.2 (0x0808) Length: 60 Handshake Protocol: Certificate Handshake Type: Certificate (11) Length: 60					
15V1.2 Record Leger: Handshake Protocol: Se ISV1.2 Record Leger: Handshake Protocol: Ce Content Type: Handshake (22) Version: TLS 1.2 (200808) Length: 667 Handshake Protocol: Certificate Handshake Protocol: Certificate (1) Length: 669 Certificates Length: 660					
LSV1.2 Record Leyer: Handshake Protocol: Se LSV1.2 Record Leyer: Handshake Protocol: Ce Content Type: Handshake (22) Version: TLS 1.2 (0x0808) Length: 60 Handshake Protocol: Certificate Handshake Type: Certificate (11) Length: 60					
LSUL2 Record Layer: Handbake Protocol: Ce Content Type: Handbake Protocol: Ce Content Type: Handbake (22) Version: TL5 1.2 (0x0000) Length: 640 Handbake Protocol: Certificate Handbake Type: Certificate (11) Length: 640 Certificates Length: 650 Certificates Length: 657	rtificate	∙.at-localitywame-Diegem,id-at-st	ateOrProvinceName=Beigi	um,id-at-commo	onlane <mark>-curs.€C steven.180</mark> ,14-8t-organizationalunitkame-TAC,14-8t-organizationkame-cisco,14-8t-countryka
LSUL2 Record Luper: Handbake Protocol: Ce Content Type: Handbake Protocol: Ce Content Type: Handbake (22) Version: TL5 1.2 (0x000) Length: 640 Handbake Protocol: Certificate Handbake Type: Certificate (11) Length: 640 Certificates Length: 640 V Certificates Length: 657	rtificate	i-st-localitywame=Diegem,id-st-st	ateOrProvinceWame=Belgi	um,id-at-commo	onWame= <mark>cour={C.steven.la0</mark> ,1d-at-organizationalunitName=TAC,1d-at-organizationName=Cisco,1d-at-countryNa
<pre>LSA1.2 Record Layer: Handbake Protocol: c Evol.2 Record Layer: Handbake (22) Version: TS.1.2 (008208) Length: 647 Handbake Protocol: Certificate Handbake Type: Certificate (11) Length: 643 Certificates Length: 640 Certificates Length: 640 Certificates: 10402040020140040000 Certificate: 104020400020140040000 Certificate: 104020400000 Vertificate: 104020400000 Vertificates: 1040204000000 Vertificates: 1040204000000 Vertificates: 104020400000000000000000000000000000000</pre>	ntificate 10202107470ee52271e3d1346 (id	∙at-localityName=Diegem,id-at-st	ateOrProvinceName-Belgi	um,id-at-commo	onWame= <mark>cgrm.fC.steven.la8</mark> ,16-at-organizationalUnitName=TAC,1d-at-organizationWame=Cisco,1d-at-countryNa
<pre>List.2 Record Luper: Handshake Protocol: ce USU.2 Record Luper: Handshake Protocol: ce Content Type: Handshake (22) Version: 15.1.2 (0x0400) Length: 647 Handshake Type: Certificate Handshake Type: Certificate (640 Vertificate) (640 Verta) Certificate: (640 Verta) Certificate: 104020430400214004000 v signedcrificate version: V3 (2) serialumber: 0x720e66221e101</pre>	ntificate 10202107470ee52271e3d1346 (id	i-st-localitywame=Diegem,id-st-st	ateOrProvinceName=Belgi	um,id-at-commo	onWame <mark>-cucm-IC.stevern.180</mark> ,16-at-organizationalUnitWame=TAC,16-at-organizationWame=Cisco,16-at-countryWa
<pre>LSA1.2 Record Layer: Handbake Protocol: c EV31.2 Record Layer: Handbake Protocol: c Content Type: Handbake (22) Version: TLS.1.2 (000303) Length: 647 Handbake Protocol: Certificate Handbake Type: Certificate (11) Length: 643 Certificates Length: 640 Vertificate: 1002026100214003020 vertificate: 1002020100214003020 vertificate: 10020201001 vertificate: 10020201001 vertificate: 10020201001 vertificate: 100202010000 vertificate: 10020201000 vertificate: 1002020000 vertificate: 10020000000 vertificate: 1002000000 vertificate: 100200000000 vertificate: 100200000000000000000000 vertificate: 10020000000000000000000000000000000000</pre>	ntificate 10202107470ee52271e3d1346 (id	-at-localitywame-Diegem,id-at-st	ateOrProvinceName-Belgi	um,id-at-commo	onWame= <mark>cucn={C.steven_188</mark> ,16-at-organizationalUnitName=TAC,16-at-organizationName=Cisco,16-at-countryNa
<pre>LSAL2 Record Layer: AndSake Protocol: c SUL2 Record Layer: AndSake Protocol: c Content Type: AndSake (22) Version: ISL 2 (0004200) Length: 647 HandSake Type: Certificate HandSake Type: Certificate (11) Length: 648 Certificates Length: 640 Vertificates Length: 640 Vertificates: DesDation20214003021 Vestificates: DesDation20214003021 Vestificates: DesDation20214003021 Vestificates: DesDation20214003021 Vestificates: DesDation20214003021 Vestificates: DesDation20214003021 Vestificates: DesDation20214003021 Vestificates: DesDation20214003021 Vestificates: DesDation20214003021 Vestificates: DesDation202140030214003021 Vestificates: Construction20214003021 Vestificates: Construction20214003021 Vestificates: Construction20214003021 Vestificates: Construction20214003021 Vestificates: Construction2021400300300300 Vestificate: Construction2000000000000000000000000000000000000</pre>	rtificate 10202107470ee62271e3d1346 (id 346150946f0a3Df1d				
<pre>LSG1.2 Record Layer: HandShake Protocol: cs USU.2 Record Layer: HandShake (22) Version: TLS 1.2 (000208) Length: 647 HandShake Type: (ertificate (11) Length: 647 HandShake Type: (ertificate (14) Certificates Length: 640 Vertificates (1648 bytes) Certificate: 10020263020214003020 v SignedCertificate version: v3 (2) serialNumBer: 0x720ee62271e301 > igsure (ecds-w15w3454) v SignedVert(0) s</pre>	rtificate 10202107470ee62271e3d1346 (id 346150946f0a3Df1d				onkame= <mark>cucm=EC.steven_188</mark> ,16=at-organizationalunitkame=TAC,16=at-organizationkame=Cisco,16=at-countryNa izationalunitkame=TAC,16=at-organizationkame=Cisco,16=at-countryName=EE)
<pre>LSG1.2 Record Layer: ManShake rotocol: s Content Type: ManShake (22) Version: 15.1.2 (ResP80) Length: 647 HanShake Type: Certificate HanShake Type: Certificate (11) Length: 648 Certificates Length: 640 Vertificates L</pre>	rtificate 10202107470ee62271e3d1346 (id 346150946f0a3Df1d				
<pre>LSA1.2 Record Layer: HandShake Protocol: 05 Content Type: HandShake (22) Version: 15.1.2 (008280) Length: 647 HandShake Protocol: Certificate HandShake Protocol: Certificate HandShake Type: Certificate (11) Length: 649 Vertificates Length: 640 Vertificate: 104020402014 Vertificate: 104020402014 Vertificate: 104020402014 Vertificates: 104020404 Vertificates: 104020404 Vertif</pre>	rtificate 10202107470ee62271e3d1346 (id 346150946f0a3Df1d				
<pre>LSG1.2 Record Layer: MadSake Protocol: ce Content Type: MadSake (22) Version: TS.1.2 (000-020) Length: 647 HandShake Type: Certificate HandShake Type: Certificate (11) Length: 648 Certificates Length: 640 Certificates Length: 640 Certificate</pre>	rtificate 10202107470ee62271e3d1346 (id 346150946f0a3Df1d				
<pre>LSAL2 Record Layer: MadSake Protocol: 6 VSUL2 Record Layer: MadSake Protocol: C Content Type: MadSake (22) Version: TSL 2 (000400) Length: 647 HandSake Type: Certificate HandSake Type: Certificate (11) Length: 640 Certificates Length: 640 Certificates Length: 640 Certificates Length: 650 Certificates Length: 650 Certificates Length: 650 Certificates Length: 650 Certificates Indonesion Version: 10(2) SecialMonter (007-Nee6227LeNDI SecialMonter (0) Handree (0) Handre</pre>	rtificate 10202107470ee62271e3d1346 (id 346150946f0a3Df1d				
<pre>List.2 Record Lupyr: Handbake Protocol: S US:1.2 Record Lupyr: Handbake Protocol: C Contert Type: Handbake (22) Version: 15.1.2 (000800) Length: 647 Handbake Type: Certificate (11) Length: 647 Vertificates Length: 640 Vertificates Length: 640 Vertificate: 1002026100214000000 v SignedCertificate Version: 30 (20) serialNumBer: 0x7300002271e001 > signedVertificate (0x7300000000000000000000000000000000000</pre>	rtificate 10202107470ee62271e3d1346 (ic 346150946f0a3Df3d ocalityname-Diegen,id-at-stateo				
<pre>Lisy.l Record Lupy:: HandSake Protocol: 6 Content Type:: HandSake (22) Version: 15.1 L2 (0mb20) Length: 667 HandSake Protocol: Certificate (11) Length: 667 HandSake Type: Certificate (14) Certificates Length: 660 Certificate: Length: 670 Certificate: Length: 660 Certificate: Length: 670 Certificate: Length: 670 Cerificate: Length: 670 Certificate: Length: 670 Cert</pre>	rtificate 10202107470ee62271e3d1346 (id 146159546f0a35f1d ocalityname-Diegem,id-at-stateo entifier)				
<pre>LSAL2 Record Layer: Anshake rotocol: se USAL2 Record Layer: Anshake rotocol: ce Content Type: Anshake (22) Version: ISL 2 (0ex080) Length: 647 Handshke type: Certificate Handshke type: Certificate (11) Length: 647 Handshke type: Certificate (14) Certificate: JeaDaddaba214ebb202 Certificate: JeaDaddaba2214ebb202 v signature: (05%-Verbea221ebb1) > Stension: (15%-Verbea221ebb1) > Extension: (16%-Verbea221ebb1) > Extension: (16%-Verbea221ebb1) > Extension: (16%-Verbea221ebb1) > Extension: (16%-Verbea221ebb1) > Extension: (16%-Verbea221ebb1)</pre>	rtificate 10202107470ee62271e3d1346 (id 1461b9946f0a3bf1d ocalitywame-Oiegem,id-at-stateO entifier) ints)				
<pre>List.2 Record Lupyr: HandSake Protocol: 5 US1.2 Record Lupyr: HandSake Protocol: C Content Type: HandSake (22) Version: 15.1 L2 (Reb200) Length: 647 HandSake Protocol: Certificate (11) Length: 649 Certificates Length: 640 Certificates Length: 640 Certificate: Joseful Content (2000) Certificate: Joseful Content (2000) Sectificate: Joseful Content (2000) Certificate: Joseful Content (2000) Sectificate: Joseful Content (2000) Sectificate: Joseful Content (2000) Sectificate: Jestificate Sectificate: Jestificate: Jestificate: Jestificate Sectificate: Jestificate: Jestificate Sectificate: Jestificate: Jestificate: Jestificate Sectificate: Jestificate: Jestificate Sectificate: Jestificate: Jestificate: Jestificate Sectificate: Jestificate: Jestificate: Jestificate Jestificate: Jestificate: Jest</pre>	rtificate 10202107470ec62273e3d3346 (id 346150946f033bf3d ocalityname=Diegem,id=at=stateO entificer) ints; me)				
<pre>List_l Record Lupyr: Handbake Protocol: 0 Content Type: Handbake Protocol: 0 Content Type: Handbake (22) Version: 15.1.2 (0mb08) Length: 647 Handbake Type: Certificate Handbake Type: Certificate (11) Length: 647 Handbake Type: Certificate (14) Certificate: (0mb17:640 Certificate: 1000200300000214000000 V signedcrificate version: 13 (2) Serializationer: 0n720002214000000 V signedcrificate version: 10(2) Serializationer: 0n720002214000000 V signedcrificate version: 10(2) Serializationer: 0n7200000000 V signedcrificate Version: 10(2) Serializationer: 0n720000000 V signedcrificate Version(10:-c-totycuage) D Extension (10:-c-totycuage) D Extension (10:-c-tot</pre>	rtificate 10202107470ec62273e3d3346 (id 346150946f033bf3d ocalityname=Diegem,id=at=stateO entificer) ints; me)				
<pre>LSAL2 Record Layer: MadSake rotocol: s USU.2 Record Layer: MadSake rotocol: c Content Type: MadSake (22) Version: TS.1.2 (000200) Length: 647 HandSake ryoe: Certificate HandSake ryoe: Certificate Certificates Length: 640 Certificates: Ba0200100201214003020 V Certificate: Ba0200100201214003020 V SigneCertificate V SigneCeritificate V SigneCertificat</pre>	rtificate 10202107470ee62271e3d1346 (id 3461D0946f0a30f1d ocalityname=Diegem,id=at=stateo entificer) ints; me) -ce=subjectiltyname)				
<pre>LSAL2 Record Layer: Anshake rotocol: s USAL2 Record Layer: Anshake rotocol: c Content Type: Anshake (22) Version: ISL 2 (000408) Length: 667 Handshake Type: Certificate Handshake Type: Certificate (11) Length: 667 Handshake Type: Certificate (12) Certificate: (0004000000000000000000000000000000000</pre>	rtificate 10202107470ee62271e3d1346 (id 3461b9946f0a3bf1d ocalityname=Olegem,id-at-stateO entifier) ints) me) -ce-subjectAltwame)				
<pre>List.2 Record Lupyr: Handbake Protocol: cs Usyl.2 Record Lupyr: Handbake Protocol: cs Content Type: Handbake (22) Version: 15.1 L2 (000830) Length: 647 Handbake Type: Certificate (11) Length: 647 Certificates Length: 640 Certificates Length: 640 Certificate: Ba0200300214000402 V Certificate: Ba0200300214000402 V Certificate: Ba0200300214000402 V Certificates Length: 640 V Certificate: Ba0200300214000402 V SigneCertificate V SigneCertificate</pre>	rtificate 10202107470ee62271e3d1346 (10 946109946f0a30f3d ocalityname-Diegen,id-at-stateo entifiar) int5) 5-ce-subjectAltName) 305				
<pre>LSAL2 Record Layer: Anshake rotocol: s USAL2 Record Layer: Anshake rotocol: c Content Type: Anshake (22) Version: ISL 2 (000408) Length: 647 Handshake ryotcol: Certificate Handshake</pre>	rtificate 10202107470ee62271e3d1346 (10 946109946f0a30f3d ocalityname-Diegen,id-at-stateo entifiar) int5) 5-ce-subjectAltName) 305				
<pre>USUL2 Record Layer: Maddake rotocol: s USUL2 Record Layer: Maddake rotocol: c Content Type: Maddake (22) Version: TLS 1.2 (000203) Length: 647 HandShake rybc: Certificate (11) Length: 647 HandShake rybc: Certificate (12) Certificates Length: 640 V Certificate: Ba0200300214003021 V Certificate: Ba0200300214003021 V Certificate: Ba0200300214003021 V SigneOcrtificate V SigneOcrtificate V</pre>	rtificate 10202107470ec62271e3d1346 (10 94610946f0a30f1d ocalityname-Diegen,id-at-stateo entifiar) int5j me) 5-ce-subjectAltName) 100 100				

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.

CISCO For Cinco	Unified Communications Solution						
	scurity · Software Upgrades · Se						
ertificate List							
Generate Self-signed	Upload Certificate/Certificate ch	ain 👔 Downio	ned CTL 🧯	Generate CSR 🔋 Download CSI	1		
tatus	0			•			
-							
 43 records found 							
Certificate List (1 - 43 of 43)						Row
ind Certificate List whe	re Certificate v begins with	1 V		Find Clear Filter 💠 🚥			
Certificate *	Common Name	Type	Key Type	Distribution	Issued By	Expiration	Description
wha	AUTHZ_cuom.steven.lab	Self-signed	RSA	cucm.steven.lab	AUTHZ_cuom.steven.lab	07/21/2038	Self-signed certificate generated by system
CalManager	cucm.steven.lab	CA-signed	RSA	cucm.steven.lab	steven-DC-CA	07/13/2022	Certificate Signed by steven-DC-CA
CaliManager-ECDSA	cucm-EC.steven.lab	Self-signed	EC	cucm.steven.lab	cucm-EC.steven.lab	02/18/2024	Self-signed certificate generated by system
CaliManager-trust	steven-DC-CA	Self-signed	RSA	steven-DC-CA	steven-DC-CA	06/01/2025	Signed Certificate
CallManager-trust	NOMAT-AD-CA	Self-signed	RSA	NOMAT-AD-CA	NOMAT-AD-CA	04/23/2028	Signed Certificate
CallManager-trust	CAP-RTP-002	Self-signed	RSA	CAP-RTP-002	CAP-RTP-002	10/10/2023	This certificate was used to sign the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC when associated with a secure pro
CallManager-trust	CAPF-eb2c64d8	Self-signed	RSA	CAPF-eb2c64d8	CAPF-eb2c64d8	04/12/2020	
CaliManager-trust	ms-AD2-CA-1	Self-signed	RSA	ms-AD2-CA-1	ms-AD2-CA-1	09/11/2024	vngtp CA
CallManager-trust	CAP-RTP-001	Self-signed	RSA	CAP-RTP-001	CAP-RTP-001	02/07/2023	This certificate was used to sign the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC when associated with a secure pro
alManager-trust	NOMAT-CA-10	Self-signed	RSA	NOMAT-CA-10	NOMAT-CA-10	08/11/2027	Signed Certificate
allManager-trust	Cisco Root CA H2	Self-signed	RSA	Cisco_Root_CA_M2	Cisco_Root_CA_M2	11/12/2037	This certificate was used to sign the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC when associated with a secure pro
allManager-trust	ACT2_SUDI_CA	CA-signed	RSA	ACT2_SUDI_CA	Cisco_Root_CA_2048	05/14/2029	This certificate was used to sign the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC when associated with a secure pro
allManager-trust	yngtp-ACTIVE-DIR-CA	Self-signed	RSA	vingtp-ACTIVE-DIR-CA	vngtp-ACTIVE-DIR-CA	02/10/2024	VNGTP-CA
CaliManager-trust	Cisco_Root_CA_2048	Self-signed	RSA	Cisco_Root_CA_2048	Cisco_Root_CA_2048	05/14/2029	This certificate was used to sign the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC when associated with a secure pro
CaliManager-trust	Cisco Manufacturing CA	CA-signed	RSA	Cisco_Manufacturing_CA	Cisco_Root_CA_2048	05/14/2029	This certificate was used to sign the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC when associated with a secure pro
CaliManager-trust	Cisco Manufacturing CA SHA2	CA-signed	RSA	Cisco_Manufacturing_CA_SHA2	Cisco_Root_CA_M2	11/12/2037	This certificate was used to sign the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC when associated with a secure pro
CallManager-trust	dccomics-WONDERWOMAN-CA	Self-signed	RSA	docomics-WONDERWOMAN-CA	dccomics-WONDERWOMAN-CA	09/19/2037	CA-byantum
CallManager-trust	CAPF-616421bc	Self-signed	RSA	CAPF-616421bc	CAPF-616421bc	07/12/2025	
CAPF	CAPF-616421bc	Self-signed	RSA	cucm.steven.lab	CAPF-616421bc	07/12/2025	Self-signed certificate generated by system
CAPF-trust	CAP-RTP-002	Self-signed	RSA	CAP-RTP-002	CAP-RTP-002	10/10/2023	This certificate was used to sign the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCM using the MIC when associated with a secure pro
CAPF-trust	CAPF-eb2c64d8	Self-signed	RSA	CAPF-eb2c64d8	CAPF-eb2c64d8	04/12/2020	
CAPF-trust	CAP-RTP-001	Self-signed	RSA	CAP-RTP-001	CAP-RTP-001	02/07/2023	This certificate was used to sign the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC when associated with a secure pro
APF-trust	Cisco Root CA M2	Self-signed	RSA	Cisco_Root_CA_M2	Cisco_Root_CA_M2	11/12/2037	This certificate was used to sign the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC when associated with a secure pro
APF-trust	ACT2_SUDI_CA	CA-signed	RSA	ACT2_SUDI_CA	Cisco_Root_CA_2048	05/14/2029	This certificate was used to sign the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC when associated with a secure pro
CAPF-trust	Cisco Root CA 2048	Self-signed	RSA	Cisco_Root_CA_2048	Cisco_Root_CA_2048	05/14/2029	This certificate was used to sign the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC installed on Cisco endpoint.
CAPF-trust	Cisco Manufacturing CA	CA-signed	RSA	Cisco_Manufacturing_CA	Cisco_Root_CA_2048	05/14/2029	This certificate was used to sign the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCM using the MIC when associated with a secure pro
CAPF-trust	Cisco Manufacturing CA SHA2	CA-signed	RSA	Cisco_Manufacturing_CA_SHA2	Cisco_Root_CA_M2	11/12/2037	This certificate was used to sign the MIC installed on Cisco endpoint. Presence of this certificate allows the end point to communicate securely with UCH using the MIC when associated with a secure pro
CAPF-trust	CAPF-616421bc	Self-signed	RSA	CAPF-616421bc	CAPF-616421bc	07/12/2025	
psec	cucm.steven.lab	Self-signed	RSA	cucm.steven.lab	cuom.steven.lab	07/12/2025	Self-signed certificate generated by system
sec-trust	cucm.steven.lab	Self-signed	RSA	cucm.steven.lab	cuom.steven.lab	07/12/2025	Trust Certificate
TLRecovery	ITLRECOVERY_cucm.steven.lab	Self-signed	RSA	cucm.steven.lab	ITLRECOVERY_cuom.steven.lab	02/14/2039	Self-signed certificate generated by system
omcat	cuom.steven.lab	CA-signed	RSA	cucm.steven.lab	steven-DC-CA	07/10/2024	Certificate Signed by steven-DC-CA
emcat-ECDSA	cucm-EC.steven.lab	CSR Only	EC	cucm.steven.lab			
emcat-ECDSA	cucm-EC.steven.lab	Self-signed	EC	cucm.steven.lab	cucm-EC.steven.lab	07/25/2023	Self-signed certificate generated by system
omcat-trust	steven-DC-CA	Self-signed	RSA	steven-DC-CA	steven-DC-CA	06/01/2025	Trust Certificate
omcat-trust	NOMAT-AD-CA	Self-signed	RSA	NOMAT-AD-CA	NOMAT-AD-CA	04/23/2028	Signed Certificate
omcat-trust	cucm-EC.steven.lab	Self-signed	EC	cucm.steven.lab	cucm-EC.steven.lab	07/25/2023	Trust Certificate
omcat-trust	cucm.steven.lab	CA-signed	RSA	cucm.steven.lab	steven-DC-CA	07/10/2024	Trust Certificate
omcat-trust	cups-EC.steven.lab	Self-signed	EC	cups.steven.lab	cups-EC.steven.lab	07/25/2023	Trust Certificate
omcat-trust	NOMAT-CA-10	Self-signed	RSA	NOMAT-CA-10	NOMAT-CA-10	08/11/2027	Signed Certificate
omcat-brust	yngtp-ACTIVE-DIR-CA	Self-signed	RSA	vngtp-ACTIVE-DIR-CA	vngtp-ACTIVE-DIR-CA	02/10/2024	Trust Certificate
omcat-trust	dccomics-WONDERWOMAN-CA	Self-signed	RSA	dccomics-WONDERWOMAN-CA	dccomics-WONDERWOMAN-CA	09/19/2037	CA Brune
VS	cucm.steven.lab	Self-signed	RSA	cucm.steven.lab	cucm.steven.lab	07/12/2025	Self-signed certificate generated by system

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.

```
<#root>
2022-07-11T19:49:01.472+02:00 vcsc traffic_server[3916]: UTCTime="2022-07-11 17:49:01,472" Module="netw
HTTPMSG:
|GET http://vcs_control.steven.lab:8443/c3RldmVuLmxhYi9odHRwcy8xMC400C4zNi4yMTUvODQ0Mw/cucm-uds/user/em
. . .
2022-07-11T19:49:01.478+02:00 vcsc traffic_server[3916]: UTCTime="2022-07-11 17:49:01,478" Module="netw
2022-07-11T19:49:01.478+02:00 vcsc traffic_server[3916]: UTCTime="2022-07-11 17:49:01,478" Module="netw
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) 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

1	Collaboration Solutions CISCO Log Analyzer	Analyzer Preview O UTC	# ?
د ۵۰۰۰	Diagnostic overvie	ew	
20		Issues found No issue Not applicable Missing Information Potential problem	
yzer	Q Search	Click on any of the below to see details or continue to analysis.	
	Result Category ^	degnostic_jog_vcsc_2022-07-11_17 33 18-DifferentCA-8443 tar.gz	
tics	 MRA (51) Configuration (39) 	Duplicate search rule for same protocol which may trigger 2 invites on the targets.	Configura
	Defects only	Centected alarms in Expressway	Configura
		Server failed to verify certificate causing TLS issues	Configura
		Certificates expired causing TLS failures and service issues	Configura
		Sentex Traffic Server Enforces Centificate Validation of UCM/IMBP/Unity nodes for MRA services [CSCwc69661]	1
		Related documentation Related defect(s) Related defect(s) CSCwc69661	
		Description	
		The tomost(-ECDSA) certificate of the following CUCM / IM&P / Unity nodes is not trusted by the Expressway-C: cucm.steven.lab, 10.48.36.215. This leads to MRA login issues.	
		Condition Expressway-C X14.2 and higher versions running MRA services are affected.	
		Further information	
		Starting with version X14.2 and higher (due to CSCwc69661), the Expressivay-C traffic server will do a TLS certificate check on the CUCM / IM8P / Unity tomcal(-ECDSA) certificates irrespective of the configuration of TLS Verify Mode se each of those servers.	it when discoverin
		Action	
		Lupdate the Expressively-C trust store with the CA certificates that signed the tomcat(-ECDSA) certificates of CUCM / IMBP / Unity nodes. Make sure that the SAN entries of the tomcat certificates contain the IP or FQDN (as shown from the log snippet below) of the respective servers how they are announced over.	
		If you are not able to update the certificates or trust store immediately, you can also apply the workaround on the CU of the Expressway-Owth the following command: xConfiguration EdgeConfigSenver VerifyOrignSenver: Off	
		Snippet	
		2022 #0-11110-10.86, 34842.08 with traffic_sever(504) [17,87] 0] MARGING for sever certificities failed worlf lights (0.84,8,323), Atlandeminate inversal fages certificate data sever-0.84,3,323(0.84, 2022 #0-1115):0(8,34842)(8) with traff_sever(504) [17,87] 0] MARGING conserver certificate worlf (0.84,3,333) and (0.84,3,323) and (0.84,3,333) and (0.84,333) and (0.84,333) and (0.84,333) and (0.8	
		depth-1 3022-07-313933)100-100+02100 vosc traffic_server[936]; [[T_NT 1] 60001 SSL connection failed for 'voor.steven.lab's error14160061SSL routines:15_process_server_certificate.verify failed	

CA diagnostic signature

Collaboration Solutions / CISCO Log Analyzer	Preview	© UTC	स्व ?
Diagnostic overvie	w		
	Issues found No issue Not applicable Missing informa	on Potential problem	
Q, Search	Click on any of the below to see details or continue to analysis.		
Result Category ^	diagnostic_log_vcsc_2022-07-11_17 49 11-CorrectCAbutwithiPor	UCM tar gz	
 MRA (51) Configuration (39) 	Duplicate search rule for same protocol which may trigger 2 in	es on the targets	Configurat
Defects only	Cetected alarms in Expressway		Configurat
	8 Server failed to verify certificate causing TLS issues		Configurat
	Certificates expired causing TLS failures and service issues		Contigura
	8 datest Traffic Server Enforces Certificate Validation of UCM/IN	IP/Unity nodes for MRA services [CSCwc69661]	3
	Related documentation	Related defect(s) CSCwc69661	
	Description The tomcat(-ECDSA) certificate of the following CUCM / IM&P / Un	nodes is not trusted by the Expressway-C: 10.48.36.215. This leads to MRA login issues.	
	Condition		
	Expressway-C X14.2 and higher versions running MRA services are Further information	flected.	
		pressway-C traffic server will do a TLS certificate check on the CUCM / IMBP / Unity torncat(-ECDSA) certificate	s irrespective of the configuration of TLS Verify Mode set when discovering
	Action		
		signed the tomcat(-ECDSA) certificates of CUCM / IM&P / Unity nodes. Ite IP or FQDN (as shown from the log snippet below) of the respective servers how they are announced over.	
	If you are not able to update the certificates or trust store immediat xConfiguration EdgeConfigServer VerifyOriginServer: Off	, you can also apply the workaround on the CU of the Expressway-Cwith the following command:	
	Snippet		
	2022-07-11719:49:01.513+02:00 vcsc traffic_server[3916]; [ET_NET: 2022-07-11719:49:01.513+02:00 vcsc traffic_server[3916]; [ET_NET:	MARSING: SNI (10.48.)6.215) not in certificate. Action=Terminate server=10.48.)6.215(10.48.)6.215) EMOGR: SSL connection failed for '10.48.)60.215': error:1416F006:SSL routine:til_process_server_certificate:cer	tificate verify failed

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 <u>CSCwa12905</u>. 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.

Related Information

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

It