



ALL IN

Troubleshooting Certificates on Aura Products

Chris Clauss - ConvergeOne

Carmen Piunno - ConvergeOne

ALL IN

Thanks for coming!

Please ask questions!

Let's make this time together worthwhile!



Why do we need certificates?

- We use TLS to provide encrypted links between servers.
- TLS encrypted links provide confidentiality protecting the communications from eavesdroppers.
- Certificates provide AUTHENTICATION. They provide the mechanism to ensure that the two parties communicating with one another are actually who they claim to be.

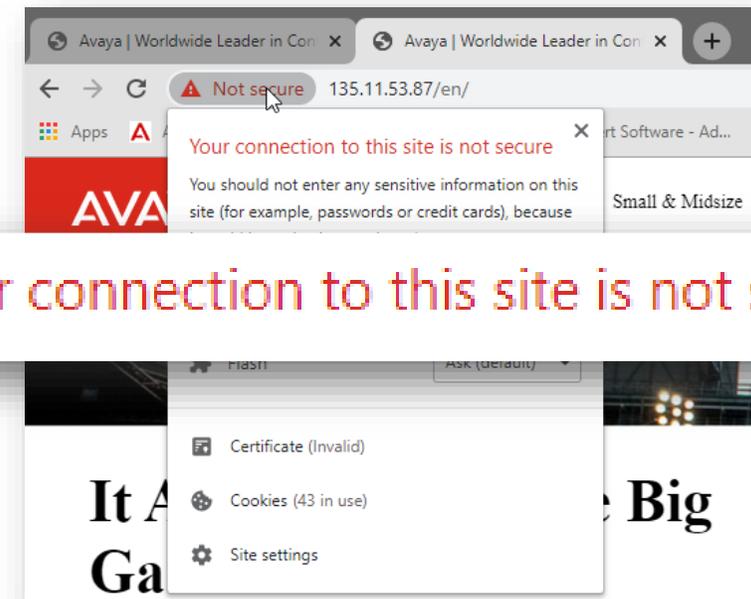
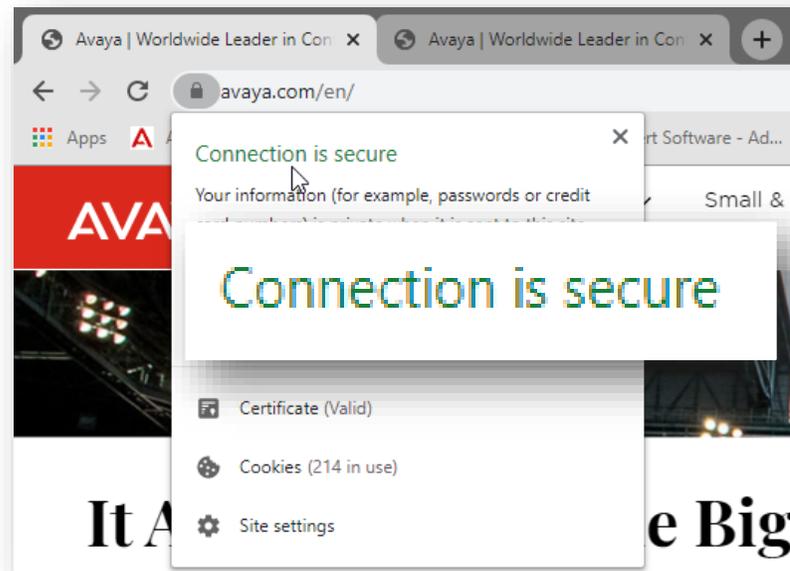


Example

- A web site that is claiming to be your secure will provide a certificate that matches the server name / domain name you are connecting to.

<https://www.avaya.com>

same as <https://135.11.53.87>

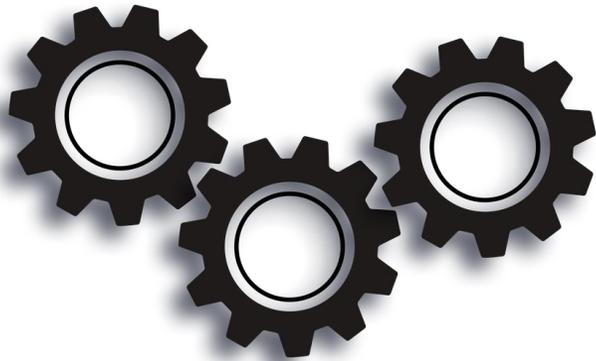




Authentication is really important, but...

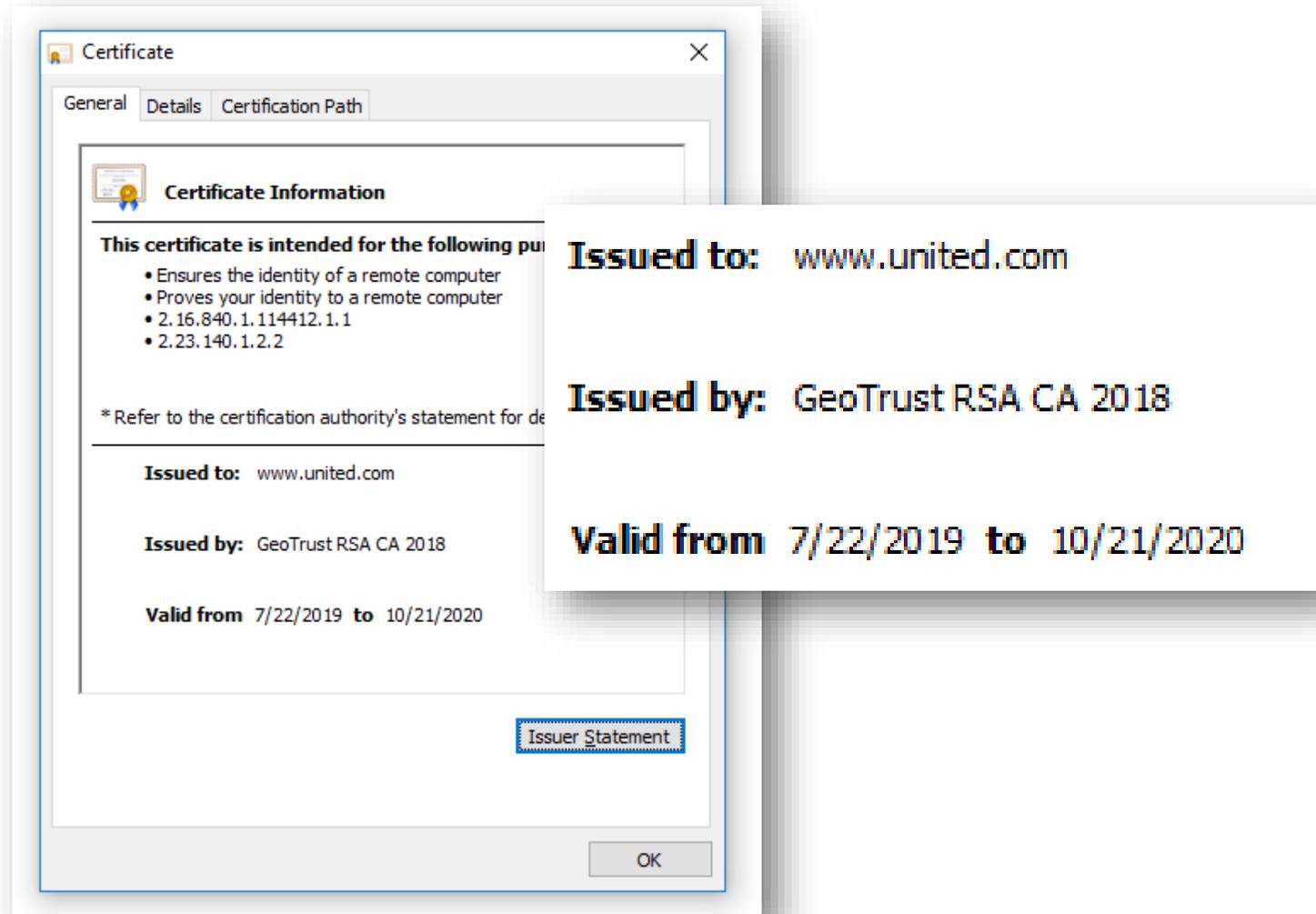
- Authentication is really important for any sites that require logins, provide secure information, etc.
- Authentication may not be that important for UC systems were server to server communications are usually “hard wired” on fixed IPs or configured to only communicate with internal servers.
- Good or bad, for TLS encryption to work, certificates are required.

A certificate includes the following



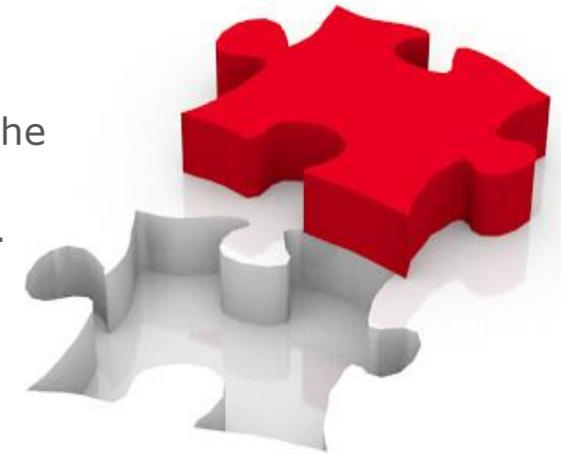
- The identity (subject) of the certificate – usually server name.
- Certificates are (almost always) signed by a third party.
- Each system that is negotiating the security will not trust the certificate unless it also trusts the signer.
- The certificate also has a validity period – best practice at this time is not more than 730 days (2years)
- The certificate will have a complexity – determined by a bit length – best practice is 2048 bits at a minimum – 4096 bits more common.

Looking at a certificate



A certificate contains two important parts

- The identity portion – usually the name of the server.
 - Identity can be defined in two places – the subject and the SAN
 - Subject field (DN) - usually the FQDN of the server – server.company.com
 - Subject Alternative Name (SAN) – the best practice location for the server identity
 - SAN may contain several entries including FQDN, short host name, IP.
- The signature portion – information on what certificate authority signed the certificate
 - The signature references the certificate authority.
 - The signer is WHAT IS TRUSTED.

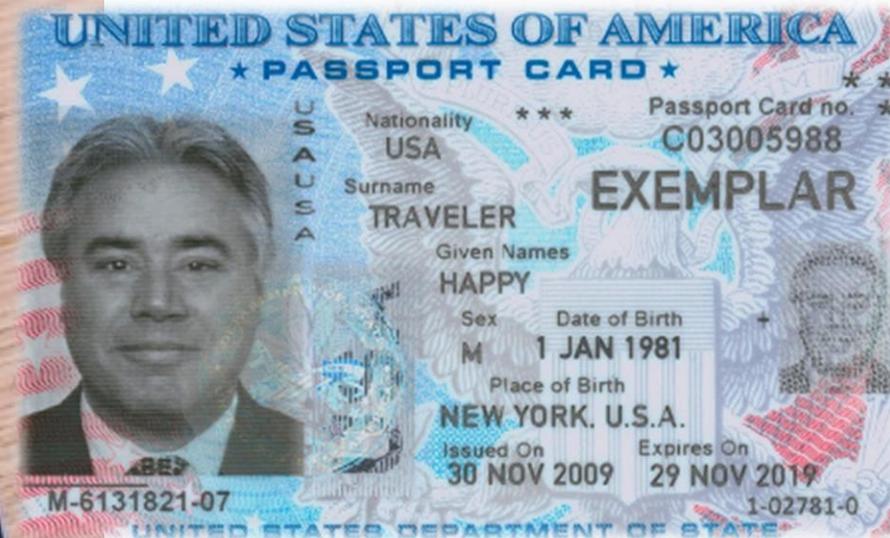


Example of trust



- Simple example – Passport
 - A passport is a certificate. It proves that you are who you say you are.
 - When you enter a country, you must provide the passport
 - The customs official does not trust you, but they trust your passport
 - Why do they trust your passport?
 - The passport is issued by a respected authority.
 - The passport contains customer security features to prevent fraud.

Here is a “real world” certificate...



- Certificate has
- An identity
- An Issuer
- A Validity Period
- Complexity

So who signs certificates?



- Self signed – example a personal check signed by you.
- Certificates signed by a certificate authority
 - Private Certificate Authorities – example is a company ID card signed by HR.
 - System Manager
 - Windows Server Certificate Authority
 - openssl on Linux
 - Public / 3rd party Certificate Authorities – example is a passport issued by a government.
 - GoDaddy
 - Verisign
 - Digicert

Self signed certificates



- A certificate that is generated by the server for itself.
- The identity of the certificate and the signer are the same.
- Some organizations consider any self signed certificate a security risk.

Private / Public Certificates



- The certificate is generated by a different server
- For private Certificate Authorities – may be System Manager or a Windows Server. Systems must import the private certificate authority “root” certificate before they will trust the identity certificate.
- For public Certificate Authorities – external company like Verisign or GoDaddy. Most operating systems trust many of these by default.
 - Public Certificate Authorities charge for generating certificates



How does a machine know to trust a certificate?

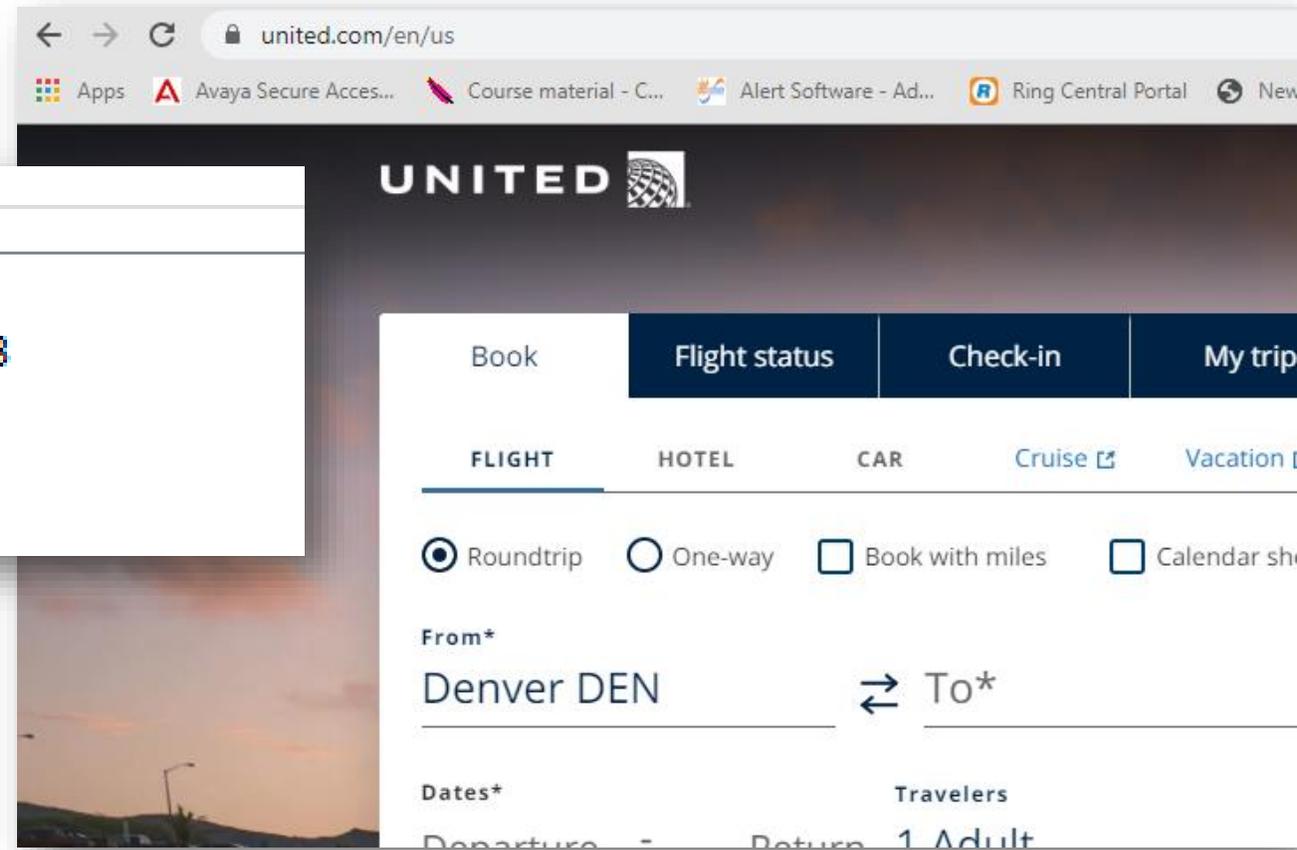
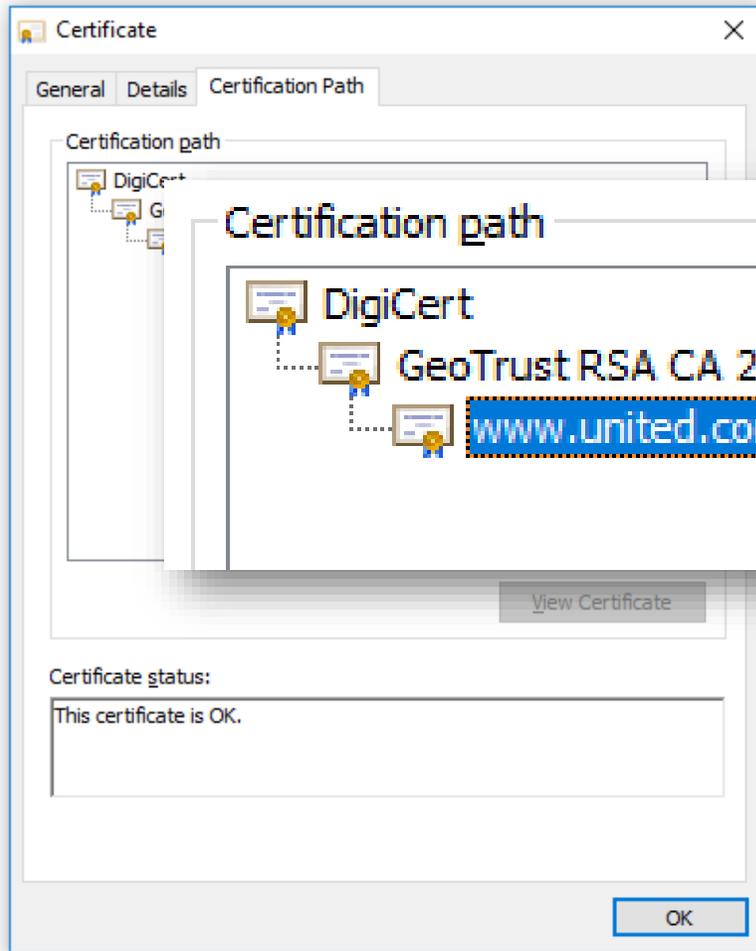
- To trust a certificate, the machine must trust the signer.
- To trust the signer, the machine must have the signers certificate installed.
 - AKA root certificate
- A signer (root) certificate is simply a certificate like every other
 - Identity portion – name of the Certificate Authority
 - Signature portion – name of the Certificate Authority
- A root certificate is just a self signed certificate that is trusted.

Certificates can have a hierarchy.

- Certificates can be signed by an “intermediate” certificate authority.
- If I trust the intermediate CA, and I trust the higher level CA, then I trust the certificate.
- Intermediate CA is used to enhance security and for ease of management of certificate requests.



Example certificate with an intermediate CA





Begin Troubleshooting Section. Where do I start?

How do I know it is a certificate problem?

- One-X softphone client blatantly telling you "the certificate I received from the server is bad."
- Maybe the link between CM and AES won't come up.
- Maybe the only information you have is that the signaling group in CM shows out of service.

What tools should I use?



Everybody troubleshoots problems differently.

- Do you spend time double checking configuration?
- Do you look at logs?
- Get your hands dirty and get a packet capture?
- Everybody's methods are going to be different here, there is no right or wrong answer.

Inevitably the only difference is that one may take less time to get to the solution.

Some tools you will use.

- Wireshark – open source tool for analysis of packets
- Windows or openssl to view certificates details
- Notepad.exe or vi to view the actual ASCII text
- openSSL – toolbox to test connection to a server and view certs
- traceSM / tracesbc – built in tool for tracing TLS
- tshark / tcpdump – test based version of Wireshark to capture packets



Preface to Understanding TLS Standards

Problem – there are no enforced rule for certificates and how they are used.

Good news – companies like Google are pushing standards.

Every product, even in Avaya can work differently.

For example, best practice wants to see certificates provide a **SAN section**.

Cert may be rejected by your client if the fqdn/ip used by the client to access the server isn't spelled out.

On the flip side - the CTI link between CM and AES has always used TLS, it has always just worked, even without a SAN.

How?

Add End Entity

End Entity Profile	INBOUND_OUTBOUND_TLS ▼	Required
Username	v-cm8	<input checked="" type="checkbox"/>
Password (or Enrollment Code)	*****	<input checked="" type="checkbox"/>
Confirm Password	*****	
E-mail address	@	<input type="checkbox"/>
Subject DN Attributes		
CN, Common name	v-cm8.clauss.org	<input checked="" type="checkbox"/>
CN, Common name		<input type="checkbox"/>
O, Organization	clauss.org	<input type="checkbox"/>
C, Country (ISO 3166)	US	<input type="checkbox"/>
OU, Organizational Unit	Lab	<input type="checkbox"/>
L, Locality	Denville	<input type="checkbox"/>
ST, State or Province	New Jersey	<input type="checkbox"/>
Other subject attributes		
Subject Alternative Name		
DNS Name	v-cm8.clauss.org	
DNS Name	v-cm8	
IP Address	172.30.0.130	
Previously added end entities	User Generated P12 file JKS file PEM file	

Steps to Troubleshooting: Step 1



Who is the client – who is the server?

Why do you think that is this important to know?

Quiz: who is the client...

- 1) sip station registering to session manager
- 2) cm to aes cti link being established
- 3) sip trunk between session manager and CM
- 4) Verint call recording registering a secure DMCC station with AES

Steps to Troubleshooting: Step 2



What service are you troubleshooting?

https web on 443

weblm on port 52233

SIP on 5061

PPM on 443

CTILINK on AES on 450

These are all protocols that can be configured to use TLS.

Why do I need to know that? This TLS negotiation happens before the actual protocol communications, and it's identical for every service, so why does this matter which service it is?

Because a server can have multiple services, on different ports, all using certificates that can be different. (See appendix)



Steps to Troubleshooting: Step 3

Now that you know server, services, port you have an idea of the identity and trusted cert. Here are 5 great places to start.

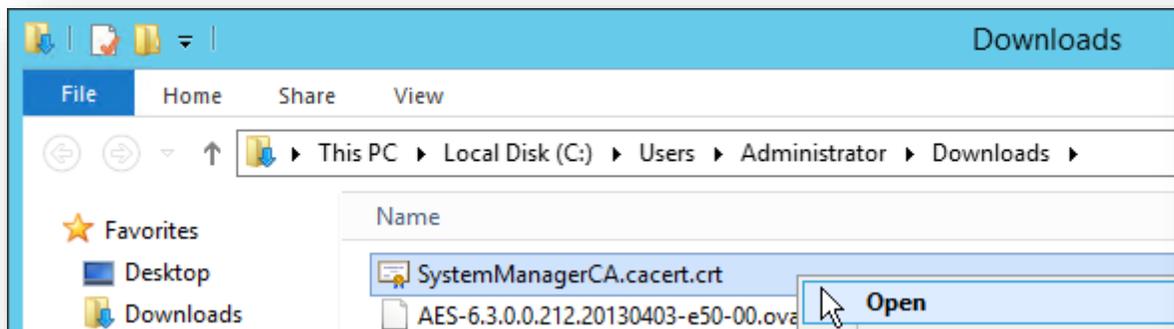
- 1) The client rejects because it does not trust the signer of the identity certificate
- 2) The client rejects because the certificate is expired
- 3) The client rejects because of missing or incorrect SAN
- 4) The servers cannot communicate because of incompatible TLS versions

Some other stuff – extended key usage – come talk to us later.

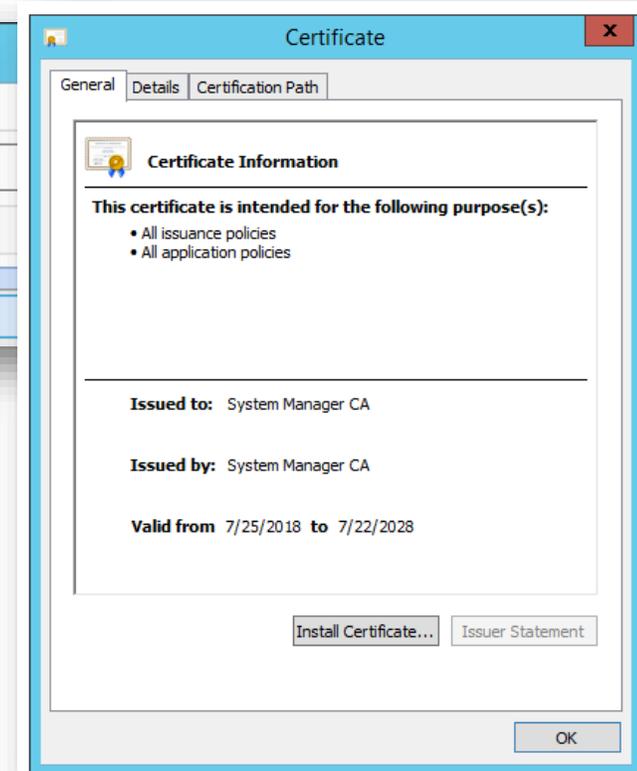
Viewing a certificate in Windows

Here is what it should look like.

Open one yourself and check it out.

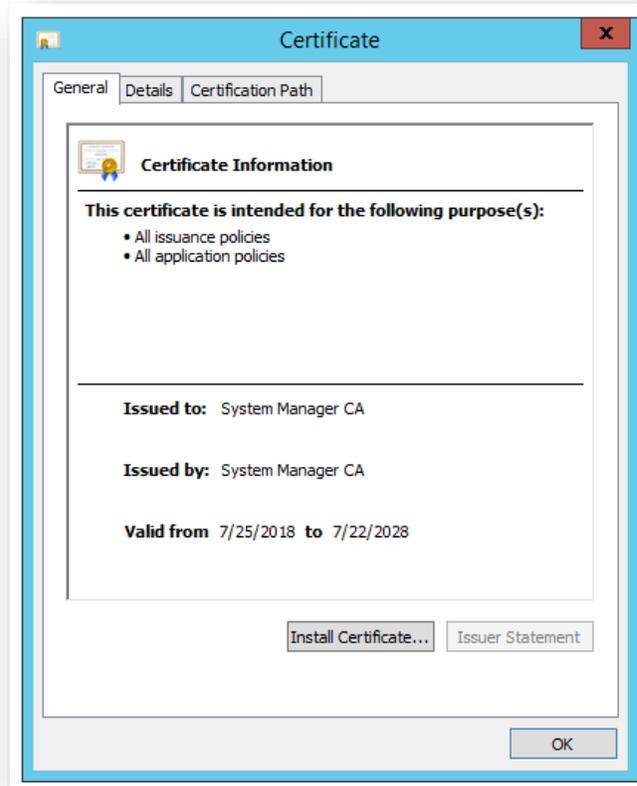


Make sure the filename can be opened by Windows – extension should be .CRT even if you downloaded it with a .PEM



Viewing a certificate in Windows

The tabs will show you details about the certificate.



This certificate is intended for the following purpose(s):

- All issuance policies
- All application policies

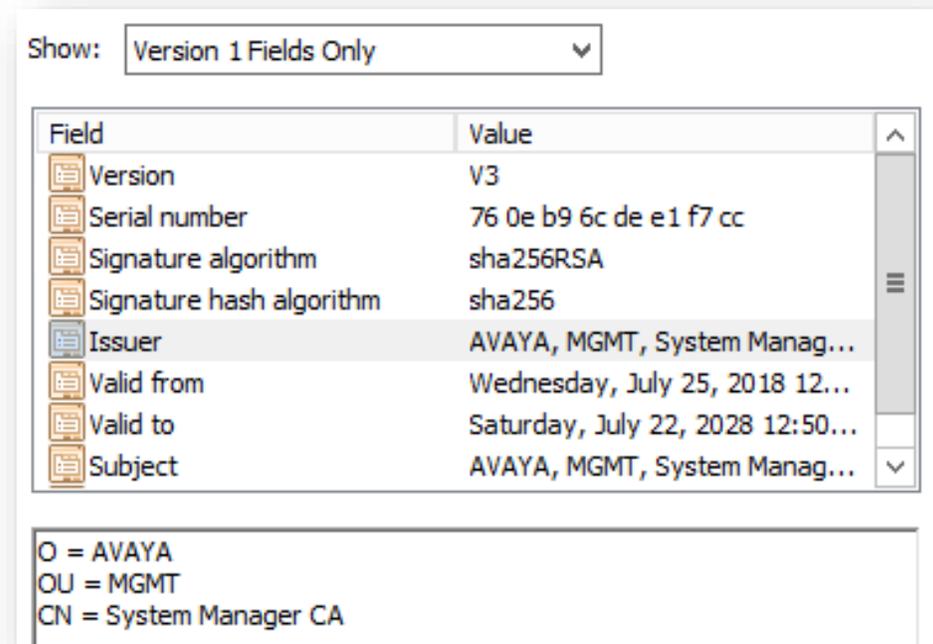
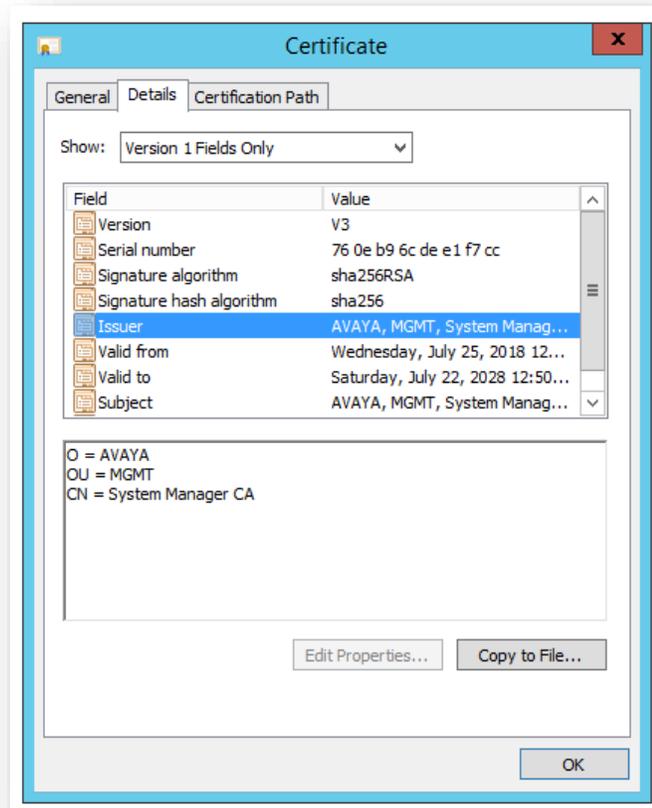
Issued to: System Manager CA

Issued by: System Manager CA

Valid from 7/25/2018 **to** 7/22/2028

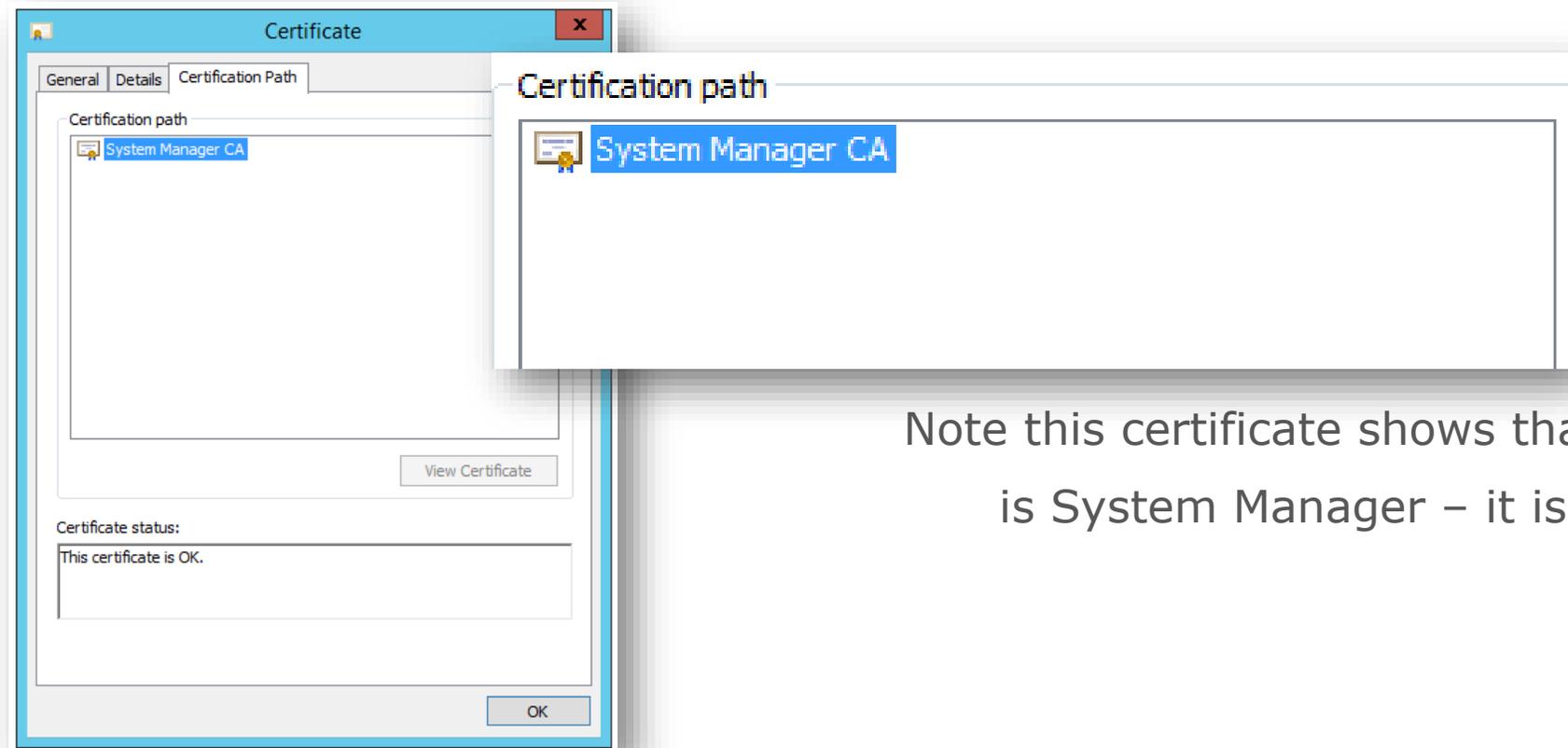
Viewing a certificate in Windows

The tabs will show you details about the certificate



Viewing a certificate in Windows

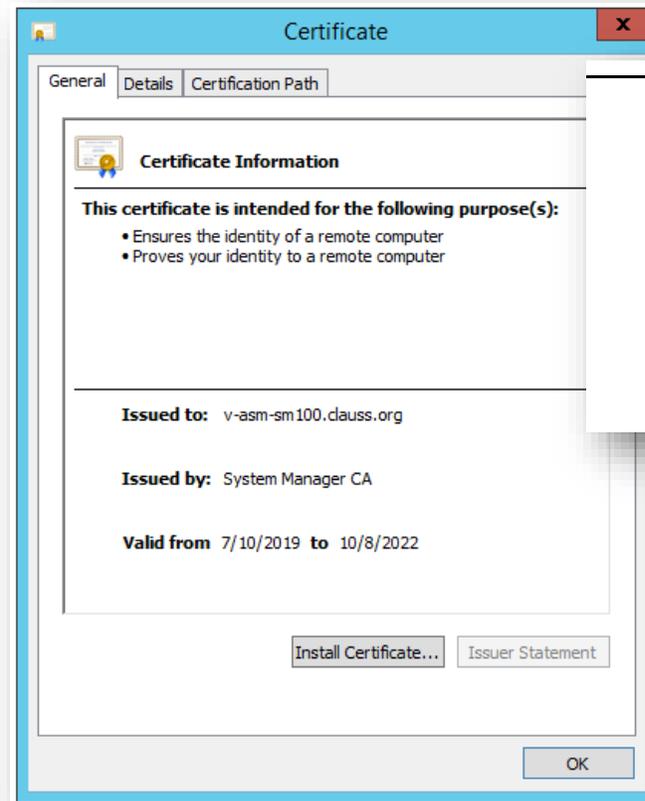
The tabs will show you details about the certificate



Note this certificate shows that the issuer is System Manager – it is self signed.

Viewing a certificate in Windows

Here is a Session Manager Certificate.



Issued to: v-asm-sm100.clauss.org

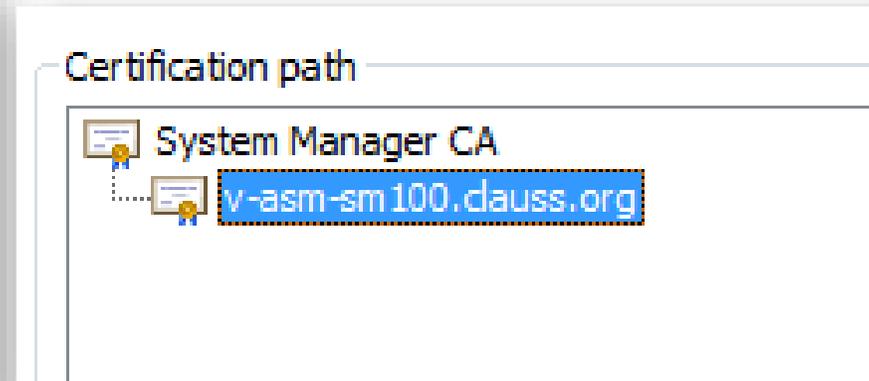
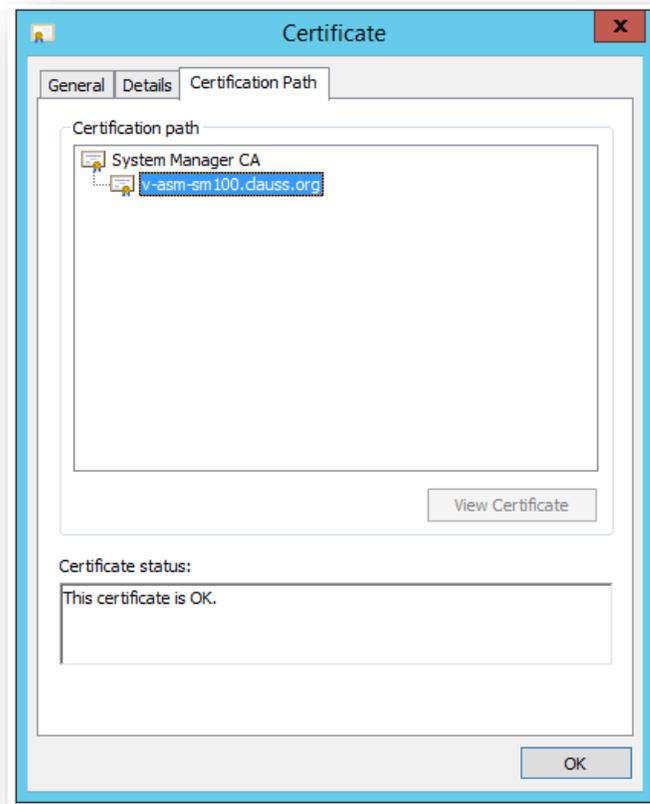
Issued by: System Manager CA

Valid from 7/10/2019 **to** 10/8/2022

Note that in this case, the issuer is SMGR

Viewing a certificate in Windows

And the certification path shows the "chain of trust"



The v-asm-sip certificate was signed by the System Manager CA, which is self signed.

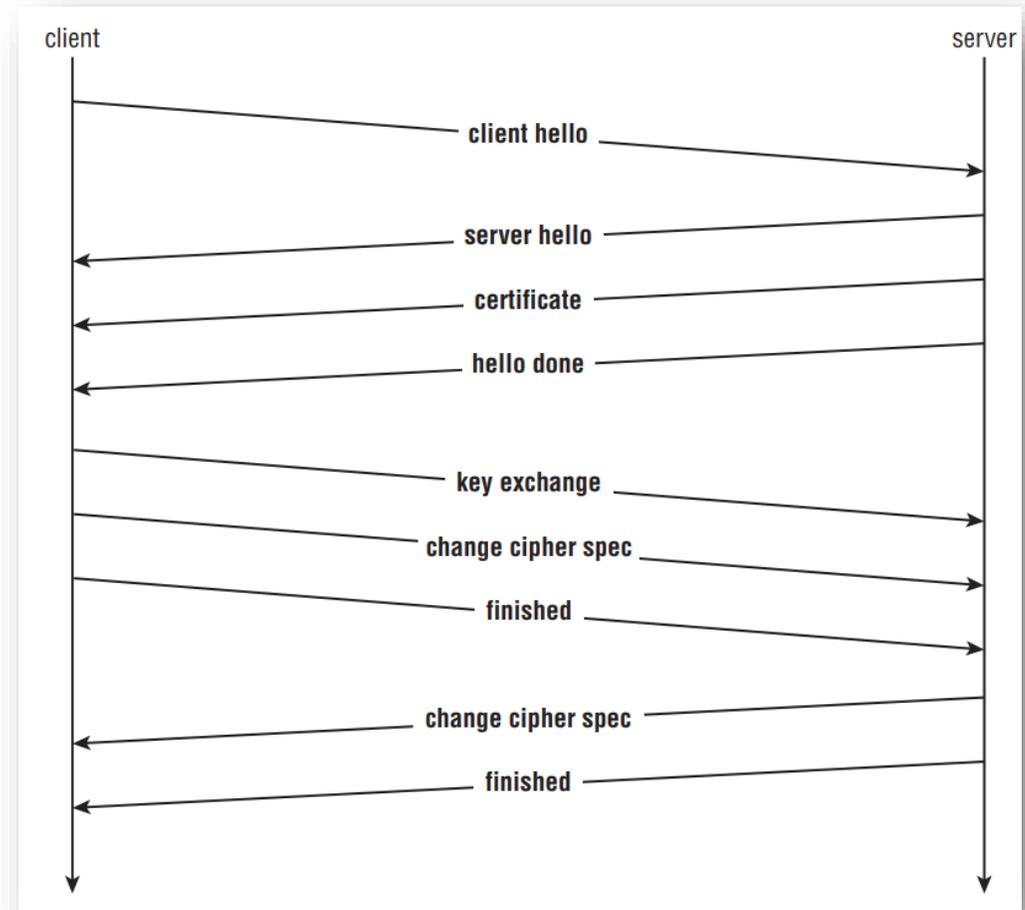


Understanding the TLS protocol

- TLS is the current standard for negotiating encryption – it is just a protocol
- TLS replaces SSL – latest version is TLS 1.2
- Can fall back to earlier versions – TLS 1.2 / 1.1 / SSL if allowed by the servers
- TLS handshake portion is clear text – not encrypted.
- After successful TLS negotiation where keys are exchanged, no further information can be viewed.

How TLS works

- The “client” computer sends a HELLO message. This message includes the version of TLS.
- The “server” computer responds with a HELLO message back. This message will be followed up with the server certificates. The server will present an identity certificate and optionally will also provide the root and intermediate certificates.



Packet Captures

So you can see that we are not lying to you, let's quickly look at a packet capture.

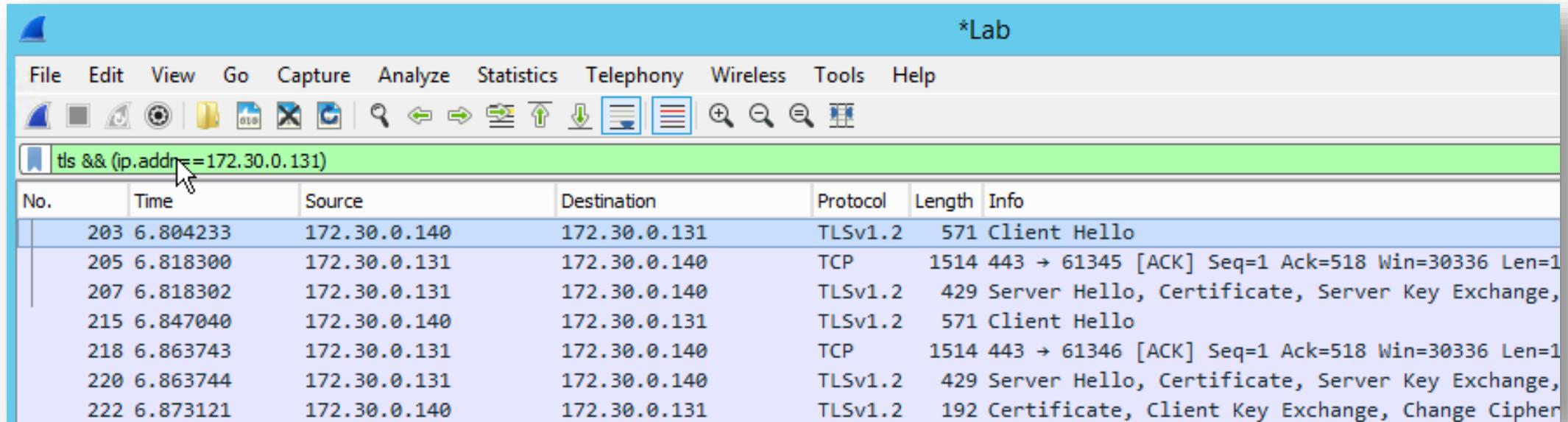
Download from – www.wireshark.org – Free!

Wireshark can capture packets or analyze packets captured on other systems.



TLS Wireshark Capture of a Successful Negotiation

Wireshark capture – filter: `tls && (ip.addr=x.x.x.x)`

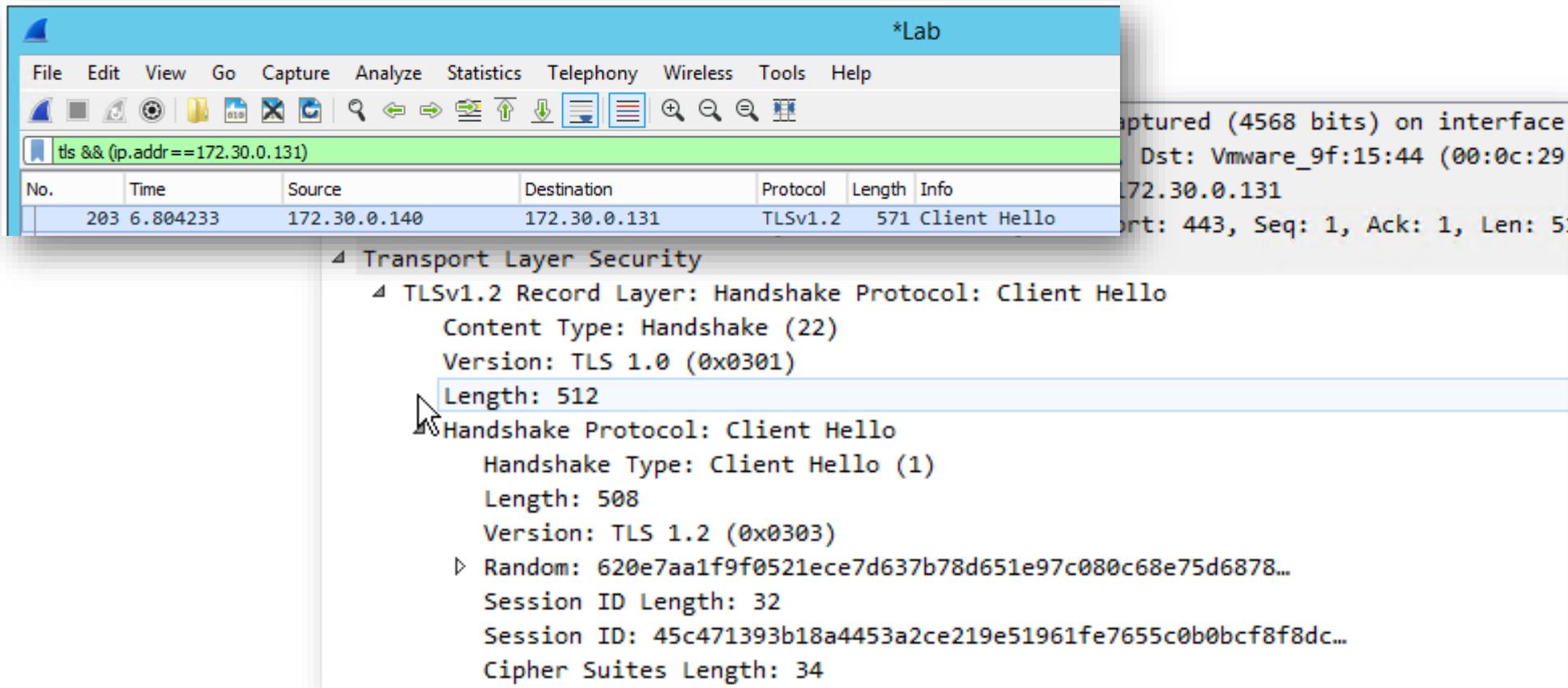


The image shows a Wireshark capture window titled '*Lab'. The filter bar contains the filter `tls && (ip.addr=172.30.0.131)`. The packet list table below shows the following entries:

No.	Time	Source	Destination	Protocol	Length	Info
203	6.804233	172.30.0.140	172.30.0.131	TLSv1.2	571	Client Hello
205	6.818300	172.30.0.131	172.30.0.140	TCP	1514	443 → 61345 [ACK] Seq=1 Ack=518 Win=30336 Len=1
207	6.818302	172.30.0.131	172.30.0.140	TLSv1.2	429	Server Hello, Certificate, Server Key Exchange,
215	6.847040	172.30.0.140	172.30.0.131	TLSv1.2	571	Client Hello
218	6.863743	172.30.0.131	172.30.0.140	TCP	1514	443 → 61346 [ACK] Seq=1 Ack=518 Win=30336 Len=1
220	6.863744	172.30.0.131	172.30.0.140	TLSv1.2	429	Server Hello, Certificate, Server Key Exchange,
222	6.873121	172.30.0.140	172.30.0.131	TLSv1.2	192	Certificate, Client Key Exchange, Change Cipher

TLS Wireshark Capture of a Successful Negotiation

client hello packet



The image shows a Wireshark capture of a TLS Client Hello packet. The packet list pane shows a single entry at time 6.804233 from source 172.30.0.140 to destination 172.30.0.131, identified as TLSv1.2 with a length of 571 bytes. The packet details pane is expanded to show the Transport Layer Security structure, including the TLSv1.2 Record Layer, Handshake Protocol, and Client Hello sub-protocol. The Client Hello sub-protocol details include a handshake type of Client Hello (1), a length of 508 bytes, and version TLS 1.2 (0x0303). It also lists random data, session ID length (32), session ID (45c471393b18a4453a2ce219e51961fe7655c0b0bcf8f8dc...), and cipher suites length (34).

No.	Time	Source	Destination	Protocol	Length	Info
203	6.804233	172.30.0.140	172.30.0.131	TLSv1.2	571	Client Hello

```
4 Transport Layer Security
  4 TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 512
  4 Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 508
    Version: TLS 1.2 (0x0303)
  4 Random: 620e7aa1f9f0521ece7d637b78d651e97c080c68e75d6878...
    Session ID Length: 32
    Session ID: 45c471393b18a4453a2ce219e51961fe7655c0b0bcf8f8dc...
    Cipher Suites Length: 34
```

TLS Wireshark Capture of a Successful Negotiation

server hello packet response – includes certificates

Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 3290

- ▶ Handshake Protocol: Server Hello
- ▾ Handshake Protocol: Certificate
 - Handshake Type: Certificate (11)
Length: 2046
Certificates Length: 2043
 - ▾ Certificates (2043 bytes)
 - Certificate Length: 1174
 - ▶ Certificate: 308204923082037aa003020102020821ebeed16136a37430... (id-at-countryName=US, Certificate Length: 863

TLS Wireshark Capture of a Successful Negotiation

Wireshark will decode the certificate information

```

└─ issuer: rdnSequence (0)
  └─ rdnSequence: 3 items (id-at-organizationName=AVAYA,id-at-organizationalUnitName=MGMT,id-at-commonName=System Manager CA)
    ├── RDNSSequence item: 1 item (id-at-commonName=System Manager CA)
    ├── RDNSSequence item: 1 item (id-at-organizationalUnitName=MGMT)
    └── RDNSSequence item: 1 item (id-at-organizationName=AVAYA)
└─ validity
  └─ notBefore: utcTime (0)
    └─ utcTime: 19-07-10 19:09:58 (UTC)
  └─ notAfter: utcTime (0)
    └─ utcTime: 22-10-08 19:09:58 (UTC)
└─ subject: rdnSequence (0)
  └─ rdnSequence: 3 items (id-at-countryName=US,id-at-organizationName=Avaya,id-at-commonName=v-smgr.clauss.org)
    ├── RDNSSequence item: 1 item (id-at-commonName=v-smgr.clauss.org)
    ├── RDNSSequence item: 1 item (id-at-organizationName=Avaya)
    └── RDNSSequence item: 1 item (id-at-countryName=US)

```



Using OpenSSL

- Without having to do a packet capture you can quickly query a server on a specific TCP port and receive back the identity certificate. Using this method you do not have to make any guesses or assumptions about which certificate it is returning. Using this information you can figure out why your link is failing.
- For example I used openssl and I can see that this server is sending back a certificate signed by Bob's House of CA, not a certificate signed by Avaya System Manager like I was told it was using.



Using openSSL

- Available on Linux servers – command line
- Command line version can also be installed on Windows.
 - One source: <https://wiki.openssl.org/index.php/Binaries>

“Simple” command line to process certificates...

```
openssl x509 -in certificate.carmen -text -noout
```

Testing connection with openssl

```
openssl s_client -showcerts -connect v-asm-sip.clauss.org:5061
```

What are the attributes in this command?

s_client command implements a generic SSL/TLS client which connects to a remote host using SSL/TLS. It is a very useful diagnostic tool.

-showcerts - displays the server certificate list as sent by the server: it only consists of certificates the server has sent (in the order the server has sent them). It is not a verified chain.

-connect host:port - specifies the host and optional port to connect to. For Session Manager sip phone registration, we use 5061 for TLS port

Testing connection with openSSL

```
openssl s_client -showcerts -connect v-asm-sip.clauss.org:5061
```

```
C:\Users\Administrator>openssl s_client -showcerts -connect v-asm-sip.clauss.org:5061
CONNECTED(00000114)
depth=1 CN = System Manager CA, OU = MGMT, O = AVAYA
verify error:num=19:self signed certificate in certificate chain
verify return:1
depth=1 CN = System Manager CA, OU = MGMT, O = AVAYA
verify return:1
depth=0 CN = v-asm-sm100.clauss.org, O = Avaya, C = US
verify return:1
---
Certificate chain
 0 s:CN = v-asm-sm100.clauss.org, O = Avaya, C = US
  i:CN = System Manager CA, OU = MGMT, O = AVAYA
-----BEGIN CERTIFICATE-----
MIIEiITCCA3GgAwIBAgIIJEx7ewdrU4owDQYJKoZIhvcNAQELBQAwOzEaMBGGA1UE
AwwRU3lzdGVtIE1hbmFnZXIgc00EXDTALBgNVBAsMSE1HTVQxDjAMBGNVBAoMBUFW
QVlBMB4XDTE5MDcxMDIxMjU0M1oXDTEyMTAwODIxMjU0M1owPjEfMBOGA1UEAwwW
di1hc20tc20xMDAuY2xhdXNzLm9yZzEOMAwGA1UECgwFQXZheWExcjZAJBgNVBAYT
A1VTMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAg3RYVwyX6Muxx1dh
U6Ly664zoC4WooIJVxBhRcz21Z3Ej704LIWlIfmKWARXR9hyH3/I+OerWNjExJJg
JT9UqE3NnaCXvfc9qawp14g2SbKQDof90KqNLYSHfHBAEMVMOIB/vzuN1jw2YRIj
7i2S/JP+xwABjnPCAsD24rb0a5xv0Cw7rGwtbxSvnVOQNbbiJSerc+PHSHttLSq
fkjKXdwP/YydrJ3kIk90tUE9PHK5vXJTOXXHVJIznBdLKP+TXx2jPxExdfVN6xAD
o4/M1zslu0rcvP/zbOrdTuHAovSDc0odiPxr0a8DaUhxH2o8E1Xr4avPjic1YTsZ
```

Testing connection with openSSL

```
openssl s_client -showcerts -connect v-asm-sip.clauss.org:5061
```

```
depth=1 CN = System Manager CA, OU = MGMT, O = AVAYA
```

```
verify error:num=19:self signed certificate in  
certificate chain
```

```
verify return:1
```

```
depth=1 CN = System Manager CA, OU = MGMT, O = AVAYA
```

```
verify return:1
```

```
depth=0 CN = v-asm-sm100.clauss.org, O = Avaya, C = US
```

```
verify return:1
```

Testing connection with openSSL

Certificate chain

0 **s**:CN = v-asm-sm100.clauss.org, O = Avaya, C = US

i:CN = System Manager CA, OU = MGMT, O = AVAYA

-----BEGIN CERTIFICATE-----

```
MIIEiTCCA3GgAwIBAgIIJEx7ewdrU4owDQYJKoZIhvcNAQELBQAwOzEaMBgGA1UE
AwwRU3lzdGVtIE1hbmFnZXIgdQ0ExDTALBgNVBAsMBE1HTVQxDjAMBgNVBAW...
```

“s” indicates that this is the subject of the certificate (the identity)

“i” indicates that this is issuer of the certificate (the CA)

Testing connection with openSSL

1 **s**:CN = System Manager CA, OU = MGMT, O = AVAYA

i:CN = System Manager CA, OU = MGMT, O = AVAYA

-----BEGIN CERTIFICATE-----

```
MIIDWzCCAkOgAwIBAgIIIdg65bN7h98wwDQYJKoZIhvcNAQELBQAwOzEaMBgGA1UE
AwwRU3lzdGVtIE1hbmFnZXIgaQ0ExDTALBgNVBAsMBE1HTVQxDjAMBgNVBAoM...
```

“s” indicates that this is the subject of the certificate (the identity)

“i” indicates that this is issuer of the certificate (the CA)

Note – since the “s”=“I” this is a SELF SIGNED certificate – normal for the root certificate.

Testing connection with openSSL

Do you want to see all of the variables in the certificate?
You can copy the text from BEGIN / END and paste it into a .txt file!

```
-----BEGIN CERTIFICATE-----  
MIIDWzCCAkOgAwIBAgIIdg65bN7h98wwDQYJKoZIhvcNAQELBQAwwOzEaMBgGA1UE  
AwwRU31zdGVtIE1hbmFnZXIgd0ExDTALBgNVBAsMIBE1HTVQxDjAMBgNVBAoM  
BUFWQV1BMB4XDTE4MDcyNTE3NTA0M1oXDTE4MDcyMjE3NTA0M1owOzEaMBgGA1UEAwwR  
U31zdGVtIE1hbmFnZXIgd0ExDTALBgNVBAsMIBE1HTVQxDjAMBgNVBAoM  
BUFWQV1BMB4XDTE4MDcyNTE3NTA0M1oXDTE4MDcyMjE3NTA0M1owOzEaMBgGA1UEAwwR  
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvA0yiotg/G9g3/zdpxcu  
Qsm0wKe4K7PLnjG9zHoGCPNDHxmX0tWxcr8F1Mq3h/ebd32E5PYnZzQt0zwB0FGL  
nkf4p4H+ZeNm26jTtpEdJGn+Rr1WnWaA1NpY/jL+j8QVr151Q61QHIpW4F9yg0mi  
7BuI154uka4Vf0ncsXcf0FcEpGKAwQDj6Mg9DSBKYRbAKpC0xctoXEiJScrSkQuC  
BwNXPr4NMwRLVSFUI2RaitSnPrTYRu9918qtS9fx20aezDAFEtVzyEKaRiC6jUws  
pRQmUo8YVr3V3caaNh3U3kNecud8IWSzB/IvzF8rfEcos980Rr8bJA/F3Y21Crik  
rQIDAQABo2MwYTAPBgNVHRMBAf8EBTADAQH/MB8GA1UdIwQYMBaAFFgyGx2IpEkT  
WyqjTAcw+1CxrTkUMB0GA1UdDgQWBRRYMhsdiKRJE1sqo0wHMPtQsa05FDA0BgNV  
HQ8BAf8EBAMCAYYwDQYJKoZIhvcNAQELBQADggEBADszqAGZqXubSGpsb6ahTxns  
Q+jC/bDBZLi+IaLF60KYfpJpLzLMs9jyE70USQKwxGbwq1SyF9e+nH5aX/Kp65oY  
LPbKIFOKZRfFEKBjSrSwkWHgcDMYA38oCaTWYD0R2pyrGBxNx9V0VIqv2cc/KS0B  
oQk77uoFvJG3By110rrkX1MG5i7bPX1SCs0QYITR8hd5icZH48SIjG1GCtBT04+8  
CVVddACpdFoaCc3u5HWnRp5zauMugWWEpbx/j32aXk+m2uMorQorSgvKXPJIDNX4  
TlXo/EWNsvye2kGA49dQQektw7wJgcSC9AVm0WktP4KwD1FA9ZsdAbItY0QAD2g=  
-----END CERTIFICATE-----
```


View Details of Certificate using Online Tool

Site will decode the certificate information for you...

```
TIXo/EWNsve2kGA49dOOektw7wlgcSC9AVm0WktP4KwD1FA9ZsdAbItY0QAD2g=  
-----END CERTIFICATE-----
```

Certificate Information:

- ✓ Common Name: System Manager CA
- ✓ Organization: AVAYA
- ✓ Organization Unit: MGMT
- ✓ Valid From: July 25, 2018
- ✓ Valid To: July 22, 2028
- ✓ Issuer: System Manager CA, AVAYA
- ✓ Serial Number: 8506940623396403148 (0x760eb96cdee1f7cc)

Testing connection with traceSM

You can also use traceSM or traceSBC to troubleshoot failures. Here is an example:

Start traceSM from Session Manager command prompt

```
[cust@v-asm ~]$ traceSM -uni
```

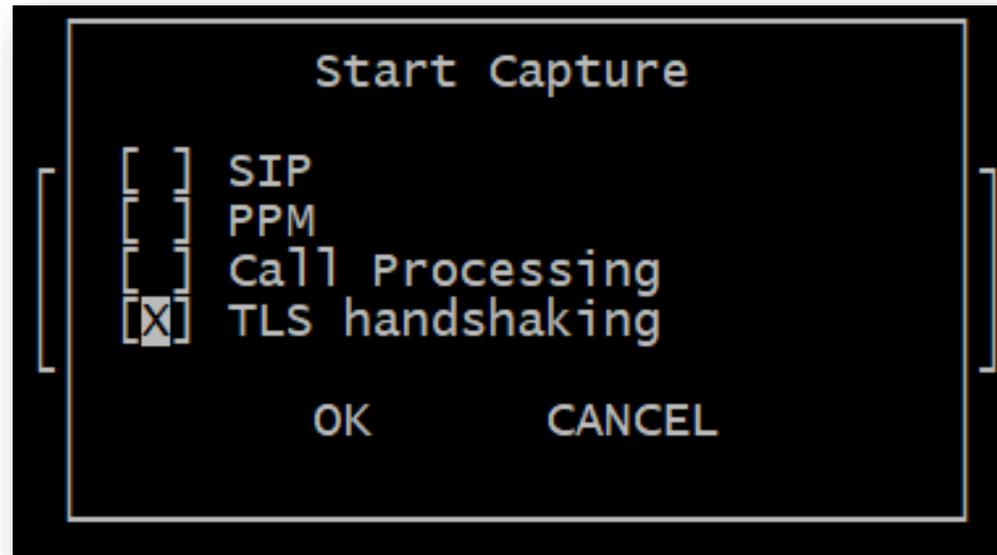
-uni option provides a better display using UNICODE characters

```
Press 's' to start the capture
```

Testing connection with traceSM



Since we are tracing TLS connections, select just TLS option...



Testing connection with traceSM

There may be quite a bit of traffic on ASM – filter IP you want to see.
Press f to filter traffic.

```
Filter Usage:
-u <URI|NUMBER> Filter calls that contain <URI|NUMBER> in
                  the 'From' or 'To' field.
-i <IP>          Filter SIP messages from/to <IP> address.
-c <CALL-ID>     Filter on the SIP 'Call-ID' header.
-s <GSID>       Filter on the SIP 'Av-Global-Session-ID' header.
-g <HEA>=<VALUE> Filter SIP packets with header <HEA> and value <VALUE>.
-o <REGEX>      Filter SIP sessions that contain <REGEX>.
-or            Use a logical OR operator instead of the implicit
              AND when using multiple filter options.
-nr           Do not display REGISTER messages.
-ns          Do not display SUBSCRIBE/NOTIFY messages.
-no          Do not display OPTIONS messages.
-ni          Do not display PING/PONG messages (RFC 3261).
-na          Do not display SM Call processing messages.
-np          Do not display PPM messages.
-nt          Do not display TLS messages.

Filter examples:
To display a call to/from 3035556666 and not REGISTER messages:
-u 3035556666 -nr
To display SIP messages from/to 1.1.1.1 and 2.2.2.2:
-i "1.1.1.1|2.2.2.2"

Current Filter: <NO FILTER>
New Filter: -i 172.30.0.52
```

Current Filter: <NO FILTER>

New Filter: -i 172.30.0.52

Example Failure – IP Phone to ASM



traceSM packet trace – filtered for specific IP address..

```
172.30.0.52      SM100
15:59:10.859    →CHello→      (T39) Client Hello
15:59:10.860    ←SHello,-     (T39) Server Hello, Certificate, Server Key Exchange, Multiple Handshake Messages
15:59:10.869    →Alert→       (T39) Fatal, Unknown CA
```

```
→CHello→
←SHello,-
→Alert→
```

```
(T1) Client Hello
(T1) Server Hello, Certificate, Server Key Exchange
(T1) Fatal, Unknown CA
```

Example Failure – IP Phone to ASM

Client Hello Packet – indicates TLS version and ciphers

```
172.30.0.52:58074 →TCP→ 172.30.0.133:5061
SSL Record Layer: Handshake Protocol: Client Hello
Version: TLS 1.0 (0x0301)
Handshake Protocol: Client Hello
Version: TLS 1.2 (0x0303)
Random
  gmt_unix_time: Jan 20, 2094 06:54:44.000000000 EST
  random_bytes: efb396b8b9819ab25a2f20e08ec8f94d3c94fcb7a474e032...
Session ID Length: 0
Cipher Suites (74 suites)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
```

Example Failure – IP Phone to ASM

Server Hello packet sent back to client

```
172.30.0.133:5061 →TCP→ 172.30.0.52:58074
TLSv1.2 Record Layer: Handshake Protocol: Server Hello
Version: TLS 1.2 (0x0303)
Handshake Protocol: Server Hello
Version: TLS 1.2 (0x0303)
Random
  gmtime_unix_time: Jul  4, 1997 17:39:24.000000000 EDT
  random_bytes: 6438b28eb310618c6c04b05111903e161310b79fc9f2971c...
Session ID Length: 0
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
Compression Method: null (0)
Extension: renegotiation_info
  Renegotiation Info extension
Extension: ec_point_formats
  Elliptic curves point formats (3)
    EC point format: uncompressed (0)
    EC point format: ansiX962_compressed_prime (1)
    EC point format: ansiX962_compressed_char2 (2)
Extension: Heartbeat
  Mode: Peer allowed to send requests (1)
```

Example Failure – IP Phone to ASM

Server Hello – Scrolling down we see the server certificate...

Certificate 1 (Identity Certificate)

```
subject: CN=v-asm-sm100.clauss.org, O=Avaya, C=US  
issuer: CN=System Manager CA, OU=MGMT, O=AVAYA
```

172.30.0.133:5061 —TCP→ 172.30.0.52:58074

Certificate 1 (Identity Certificate)

signedCertificate

serialNumber: 2615601251931083658

signature (sha256withRSAEncryption)

issuer: CN=System Manager CA, OU=MGMT, O=AVAYA

validity

notBefore: 19-07-10 21:25:42 (UTC)

notAfter: 22-10-08 21:25:42 (UTC)

subject: CN=v-asm-sm100.clauss.org, O=Avaya, C=US

subjectPublicKeyInfo (rsaEncryption)

subjectPublicKey: 3082010a0282010100ab7458570c97e8cbb1c7576153a2f2...

extensions: 9 items

Extension (id-ce-subjectAltName)

GeneralNames: 1 item

Example Failure – IP Phone to ASM

Server Hello – Scrolling down further the signing (root) certificate...

Certificate 2 (Root Certificate)

```
subject: CN=System Manager CA, OU=MGMT, O=AVAYA  
issuer: CN=System Manager CA, OU=MGMT, O=AVAYA
```

172.30.0.133:5061 —TCP→ 172.30.0.52:58074

```
Certificate 2 (Root Certificate)  
  signedCertificate  
    serialNumber: 8506940623396403148  
    signature (sha256withRSAEncryption)  
    issuer: CN=System Manager CA, OU=MGMT, O=AVAYA  
    validity  
      notBefore: 18-07-25 17:50:43 (UTC)  
      notAfter: 28-07-22 17:50:43 (UTC)  
    subject: CN=System Manager CA, OU=MGMT, O=AVAYA  
    subjectPublicKeyInfo (rsaEncryption)  
      subjectPublicKey: 3082010a0282010100bc04328a8b60fc6f60dffcd7172e...  
    extensions: 4 items  
      Extension (id-ce-basicConstraints)  
        critical: True
```

Example Failure – IP Phone to ASM



The third packet is an error message from the client...

```
172.30.0.52:58074 →TCP→ 172.30.0.133:5061
TLsv1.2 Record Layer: Alert (Level: Fatal, Description: Unknown CA)
  Version: TLS 1.2 (0x0303)
  Alert Message
    Level: Fatal (2)
    Description: Unknown CA (48)
```

That is awfully nice of them to build something into the protocol that tells you exactly what you did wrong! That is not always going to be the case.

Example Failure – IP Phone to ASM

Checking Windows, the SMGR certificate is not installed.
Here I installed it. More on root certificates later.



Example Failure – IP Phone to ASM

After installing the SMGR ROOT certificate on the Windows machine...

```
172.30.0.52
SM100
15:01:40.565  →CHello→ (T19) Client Hello
15:01:40.566  ←SHello, (T19) Server Hello, Certificate, Server Key Exchange, Multiple Hands
15:01:40.577  →Cert, C (T19) Certificate, Client Key Exchange, Encrypted Handshake Message
15:01:40.577  ←EncHand (T19) Encrypted Handshake Message
15:01:40.584  →AppData→ (T19) Application Data
15:01:40.585  ←AppData (T19) Application Data
15:01:40.593  →AppData→ (T19) Application Data
15:01:40.595  ←AppData (T19) Application Data
```

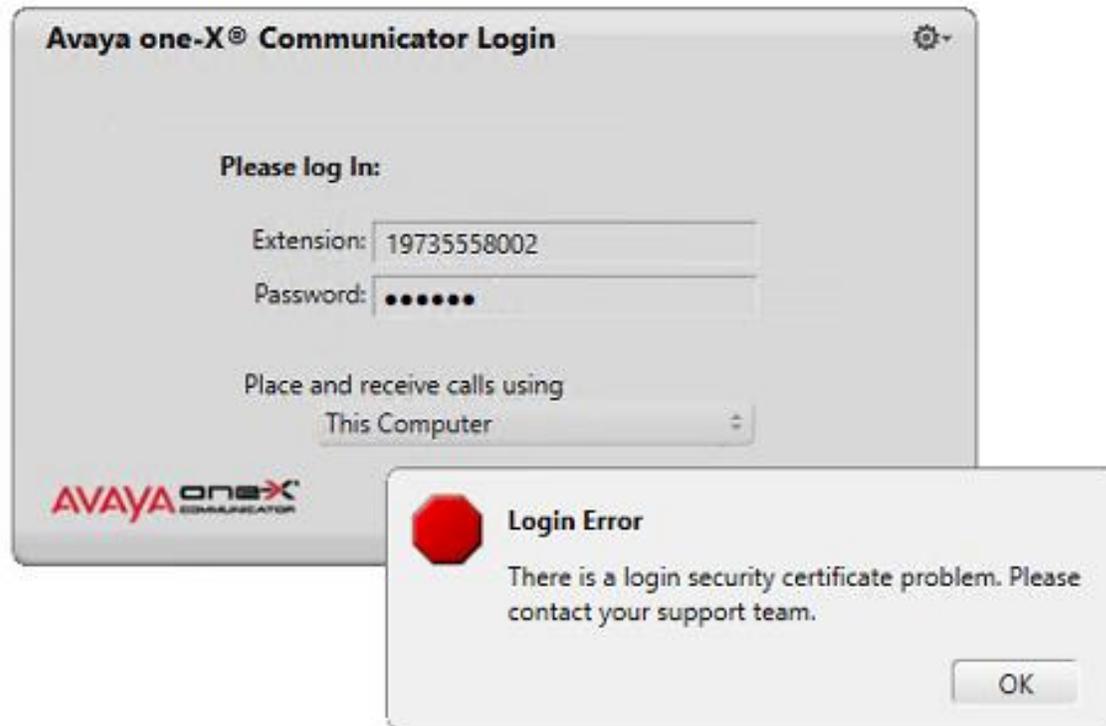
The "AppData" messages are SIP messages encrypted...

```
172.30.0.133:5061 →TCP→ 172.30.0.52:60292
TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
Version: TLS 1.2 (0x0303)
Handshake Protocol: Encrypted Handshake Message
```

Example Failure – IP Phone to ASM

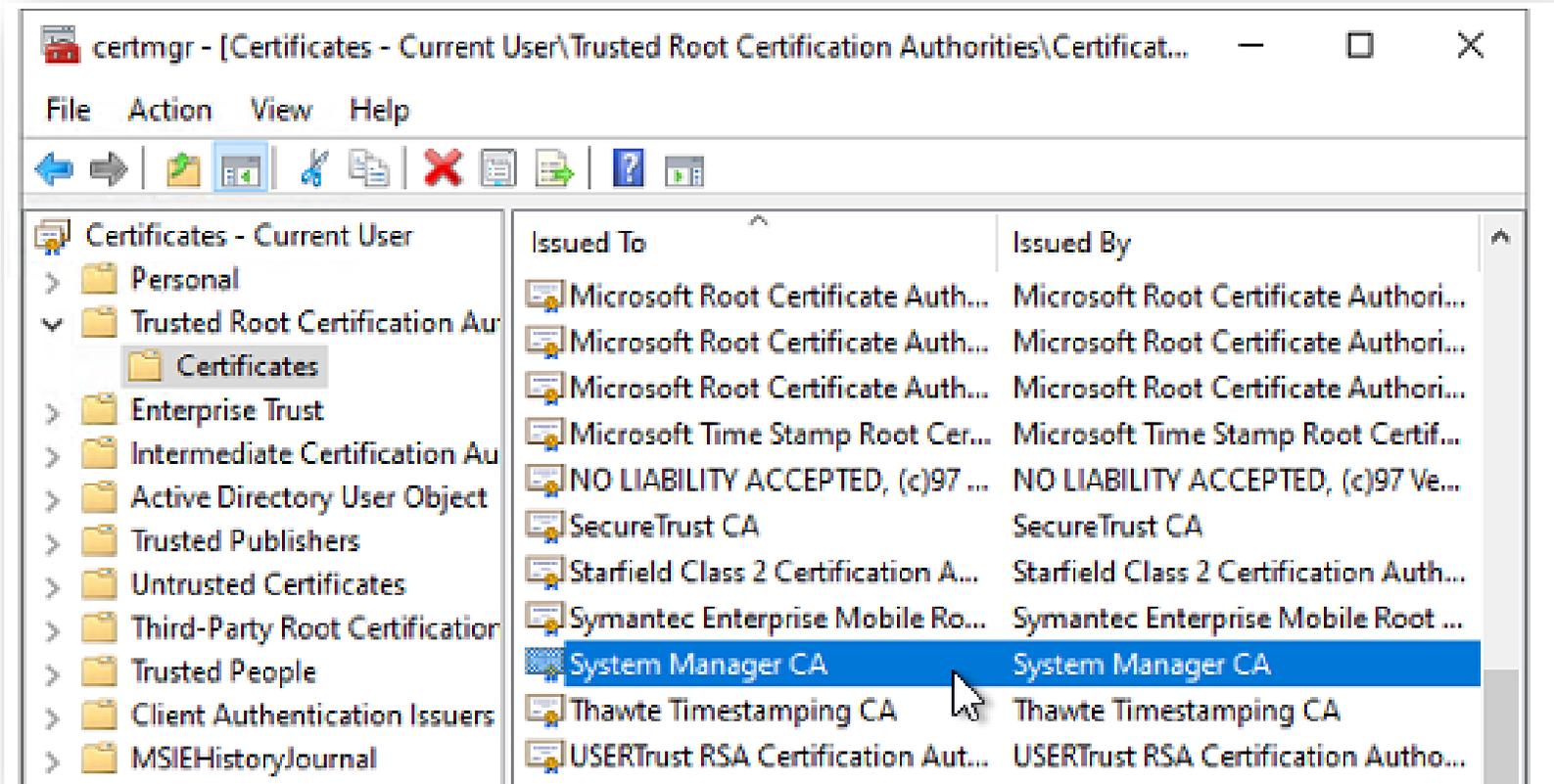


However the application still indicates that there is a problem! Oh no!



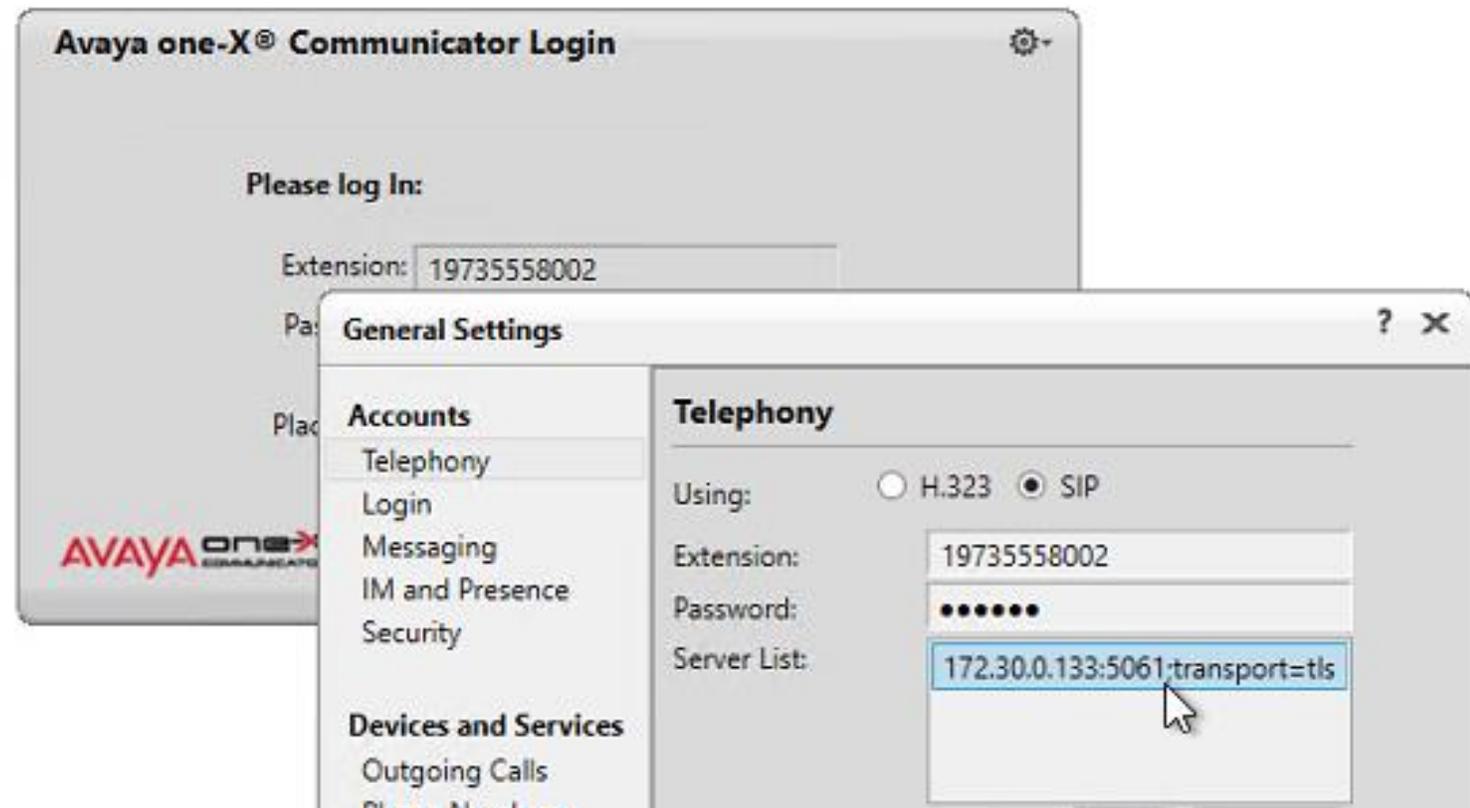
Example Failure – IP Phone to ASM

We need the System Manager cert – but it is already installed. What's up?



Example Failure – IP Phone to ASM

Let's look at the client config -



Example Failure – IP Phone to ASM



Let's look at the client config –

Client trying to connect to 172.30.0.133, but the certificate presented by Session Manager is built using only the FQDN. Which it should be! This is a SAN validation issue. Meaning: who you say you are isn't who I reached out to, therefore I am not going to trust you.

Manage Elements		Discovery	
<input type="radio"/>	spiritalias	spiritalias	spiritalias
<input checked="" type="radio"/>	securitymodule_http	securitymodule_http	securitymodule_http

Certificate Details

Subject Details	C=US, O=Avaya, CN=v-asm-sm100.clauss.org
Valid From	Wed Jul 10 17:25:43 EDT 2019
Key Size	2048
Issuer Name	O=AVAYA, OU=MGMT, CN=System Manager CA



How to resolve the SAN validation Issue

There are actually multiple options to resolve this type of certificate issue:

- 1) Modify the identity certificate in session manager so that the ip address *is* in the SAN. We are never going to do this because it is bad practice to use IP addresses in certificates and it's not actually that easy to change a certificate on a whim. More on that later.
- 2) Tell OneX to talk sm using the fqdn instead of the IP address. This would be the most proper option. However, let's pretend that I can't do this because I have no way of getting the fqdn into DNS in a timely manner.
- 3) As it turns out, I have the option to configure OneX communicator client not to care that certificate is invalid. What? That is scary!

Modify OneX to Not Validate Certificate Hostname

For Communicator we have to tell it not to validate the host name in the certificate. You may be thinking this is bad, but truthfully every software developer does this for user experience. Think about the web browser you use every day when logging into Avaya servers. It constantly tells you "I think you're being bamboozled" and you just click right through "I accept the consequences".

```
InstallConfig.xml [x]
1  <?xml version="1.0" encoding="UTF-8"?>
2  <InstallConfig xmlns="http://xml.avaya.com/endpointAPI">
3      <CitrixMode>false</CitrixMode>
4      <CultureName>en-US</CultureName>
5      <DscpForVideo>0</DscpForVideo>
6      <EnableCCEIntegration>true</EnableCCEIntegration>
7      <EnableVideo>true</EnableVideo>
8      <IsUpgrade>false</IsUpgrade>
9      <MaxTrustCerts>-1</MaxTrustCerts>
10     <SignalProtocol>2</SignalProtocol>
11     <EnableHostnameValidation>false</EnableHostnameValidation>
12 </InstallConfig>
```



Certificate Trust

Earlier We Added a root certificate to the trust store of Windows. How do I view or modify that on my Avaya server?

It varies GREATLY from product to product as you can see:

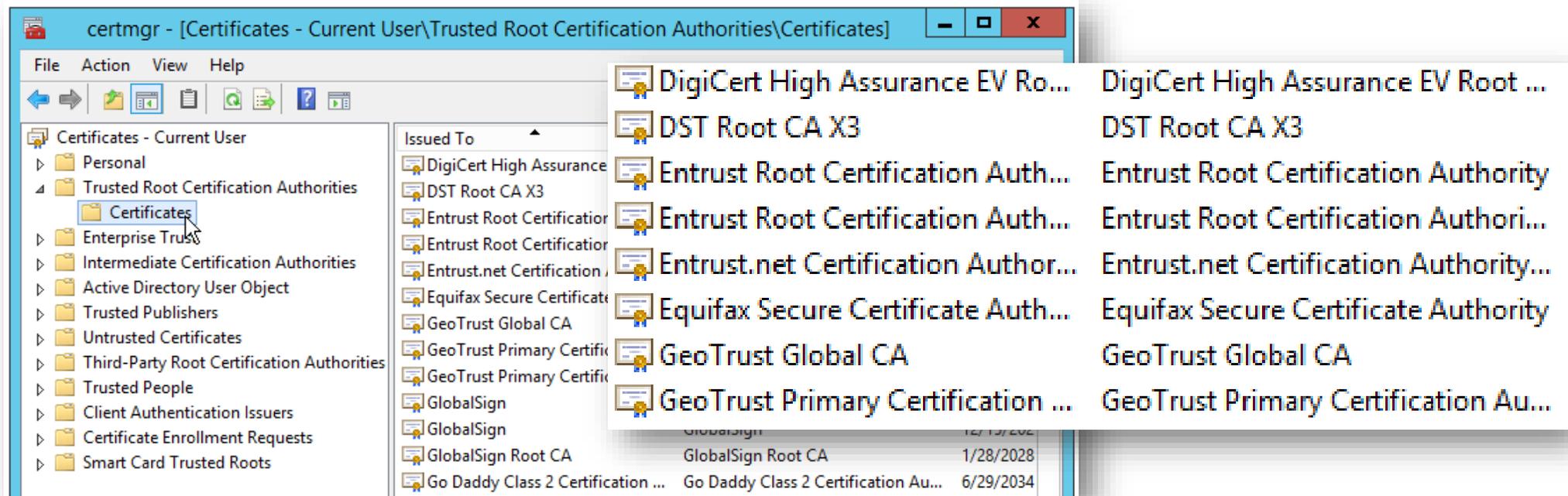
Windows. Why does my computer trust “united.com”

The image displays several overlapping windows from the Windows Certificate Manager (certmgr). The top window shows the 'Certificate Information' for a certificate issued to 'www.united.com'. The 'Certification Path' window shows a chain of trust: DigiCert -> GeoTrust RSA CA 2018 -> Certificate. The bottom window shows the 'Trusted Root Certification Authorities' list, which includes DigiCert Global Root CA, DigiCert Global Root G2, and DigiCert Global Root G3.

Issued To	Issued By	Expiration Date
DigiCert Global Root CA	DigiCert Global Root CA	11/9/2031
DigiCert Global Root G2	DigiCert Global Root G2	1/15/2038
DigiCert Global Root G3	DigiCert Global Root G3	1/15/2038

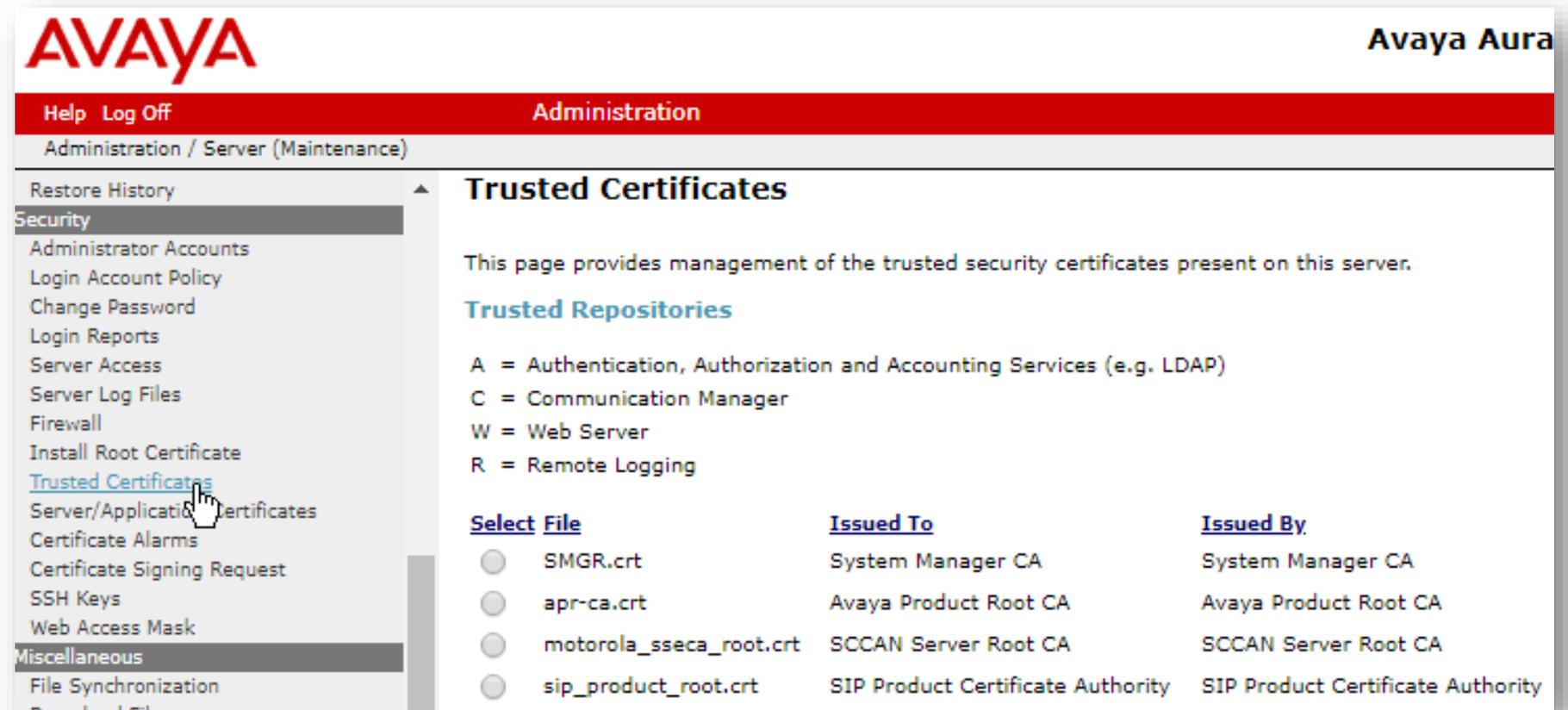
How to check what CA my systems trust?

In Windows, use certmgr.msc – Start / Run / certmgr.msc
Select Trusted Root Certification Authorities / Certificates



On other Avaya Systems – in Web interface

On CM – Admin / Server / Security / Trusted Certificates



The screenshot shows the Avaya Aura Administration web interface. The top navigation bar includes the Avaya logo, "Help Log Off", and "Administration". Below this, the breadcrumb "Administration / Server (Maintenance)" is visible. A left-hand navigation menu lists various security-related options, with "Trusted Certificates" highlighted and a mouse cursor pointing to it. The main content area is titled "Trusted Certificates" and contains the following text:

This page provides management of the trusted security certificates present on this server.

Trusted Repositories

A = Authentication, Authorization and Accounting Services (e.g. LDAP)
C = Communication Manager
W = Web Server
R = Remote Logging

<u>Select File</u>	<u>Issued To</u>	<u>Issued By</u>
<input type="radio"/> SMGR.crt	System Manager CA	System Manager CA
<input type="radio"/> apr-ca.crt	Avaya Product Root CA	Avaya Product Root CA
<input type="radio"/> motorola_sseca_root.crt	SCCAN Server Root CA	SCCAN Server Root CA
<input type="radio"/> sip_product_root.crt	SIP Product Certificate Authority	SIP Product Certificate Authority

On other Avaya Systems – in Web interface

On AES – Security / Certificate Management / CA Trusted Certificates

The screenshot displays the Avaya Application Enablement Services Management Console. The top left features the Avaya logo. The main title is "Application Enablement Services Management Console". On the top right, system information is provided: "Number of prior failed login attempts: 0", "HostName/IP: v-aes/172.30.0.134", "Server Offer Type: VIRTUAL_APPL", "SW Version: 8.1.1.0.0.8-0", "Server Date and Time: Mon Jan 2", and "HA Status: Not Configured". A red navigation bar contains "Security | Certificate Management | CA Trusted Certificates". A left sidebar lists navigation options: "AE Services", "Communication Manager Interface", "High Availability", "Licensing", "Maintenance", "Networking", "Security" (expanded), "Account Management", "Audit", "Certificate Management" (expanded), and "CA Trusted Certificates" (selected). The main content area is titled "CA Trusted Certificates" and includes buttons for "View", "Import", "Export", and "Delete". Below these buttons is a table with the following data:

Alias	Status	Issued To	Issued By
<input type="radio"/> avayaprca	valid	Avaya Product Root CA	Avaya Product Root CA
<input type="radio"/> avaya_sipca	valid	SIP Product Certificate Authority	SIP Product Certificate Authority
<input type="radio"/> serverCertDefault	valid	v-aes-560366381-labUseOnly	v-aes-560366381-labUseOnly

On other Avaya Systems – in Web interface

For System Manager elements – Services / Inventory / Elements
Select element and More Actions / Manage Trusted Certificates

The screenshot displays the 'Elements' management interface. At the top, there are buttons for 'View', 'Edit', 'New', 'Delete', 'Details', and 'Get Current Status'. Below these, it shows '2 Items' with a 'Reset' button and a 'Show All' dropdown. A table with two columns, 'Name', contains two entries: 'v-asm-sip' (checked) and 'v-asm-sip' (unchecked). A 'Select: All, None' option is at the bottom left. The 'More Actions' dropdown menu is open, listing options: 'Manage Trusted Certificates', 'Manage Identity', 'Manage Trusted Certificates', 'Manage', 'Unmanage', 'Import', 'View Notification Status', 'SAL Gateway configuration', 'Product Registration', and 'View Certificate Add Status'. A mouse cursor is pointing at the first 'Manage Trusted Certificates' option.

<input type="checkbox"/>	Name
<input checked="" type="checkbox"/>	v-asm-sip
<input type="checkbox"/>	v-asm-sip

Select : All, None

- More Actions ▾
 - Manage Trusted Certificates
 - Manage Identity
 - Manage Trusted Certificates
 - Manage
 - Unmanage
 - Import
 - View Notification Status
 - SAL Gateway configuration
 - Product Registration
 - View Certificate Add Status



What CA certificates need to be installed on your Server?

- The “root” certificates of any identity certificate that you are going to receive as a client. There is no limit to the number of root certificates that you can trust. If you are unsure which ones you need, go to town.
- The root certificates of whoever signed your identity certificate that you are shelling out for all of your services (remember each service actually uses its own certificate, even if you decided to install the same certificate for every service, this is still the case)
- The intermediate certificates of whoever signed your identity certificate as a server (more on intermediates later).
- Note: Public CA pre-trusted root certificates that are automatically trusted by every other device ARE NOT installed by default on most Avaya systems. There would be no need for this. Quiz: do you think those public ca trusts ARE actually installed somewhere in Linux?

Where do intermediate CAs come into play?



- In our troubleshooting examples I have been simplifying the chain of trust to just one level: one root ca, the one who issued the certificate. That is how your internal CA is going to be, like if you are using System Manager. However this is not how any other certificate authorities work, and this was done on purpose.
- Your phone or computer or server only pre-trusts a handful of CAs. These are referred to as the Public Root Certification Authorities, or root CAs. There is something very important to note about this: public Root CAs do not actually issue identity certificates. Think of them like offline servers inside of a locked vault that cannot be infiltrated. Instead, these root authorities only sponsor another CA and allow them to actually issue the identity certificates. This is called an intermediate or sub CA.
- These sub CAs are on the grid and have to be because they are constantly issuing certificates, thus they are more susceptible to attack. If a sub CA were to get compromised, that entire CA it can be "revoked" by the root, new CRLs are pushed too all devices, and now nobody trusts those certificates. This is assuming your server gets CRL updates. Your iOS phone probably gets them regularly. Your Avaya server, on the other hand, has no way to get them.



Why do I trust intermediate CAs?

- Because of intermediate Cas we now have a circumstance where your client only trusts GoDaddy but the certificate being presented during TLS setup was issued by Bobs CA Barn. Why did your customer chose to purchase a certificate from Bobs? Because Bobs offered the best price for certificates. For this certificate to be considered valid, your client also needs to see proof that Bobs CA Barn was eventually signed by GoDaddy.
- There is one more piece of the puzzle that we have not yet talked about:
- Along with the identity certificate sent by the server, the server is allowed to send as many more certificates as it wants. The section for certificates in the TLS process is called `certificate_list`. So, AFTER sending it's identity certificate (signed by Bobs), server sends you back another certificate. Low and behold, it is the intermediate certificate. There is nothing special about this certificate: This "intermediate certificate" is simply Bob's CA Barn's certificate. By looking at this "intermediate certificate" your client device is able to see Bobs certificate was signed by GoDaddy and now your client is happy and considers this connection validated.



Troubleshooting using Packet Capture



You can use a packet capture to troubleshoot issues as well.



Capturing packets when you don't have access

- Sometimes it is difficult to run utilities on some servers without proper access / root access.
 - I/T can do port mirrors to a PC running wireshark.
 - Network teams can sometimes capture packets at a switch

Troubleshooting using Packet Capture

AES Link will not come up. We get into AES where we have root access and we know we can easily get a packet capture.

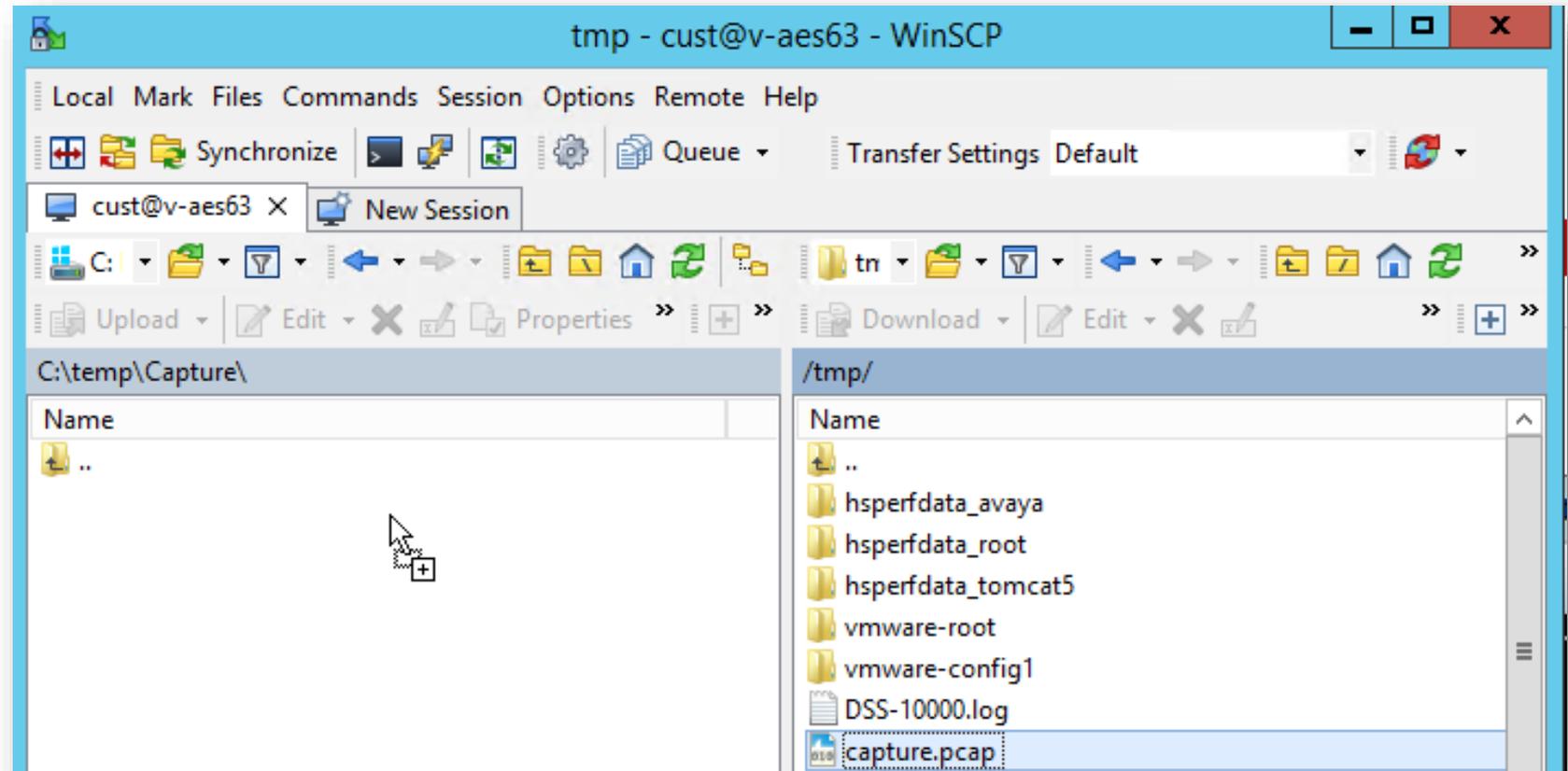
```
tcpdump -w /tmp/capture.pcap capture packets to file
```

```
[root@v-aes63 /]# tcpdump -w /tmp/capture.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
5 packets captured
5 packets received by filter
0 packets dropped by kernel
[root@v-aes63 /]# tcpdump -w /tmp/capture.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes

313 packets captured
313 packets received by filter
0 packets dropped by kernel
```

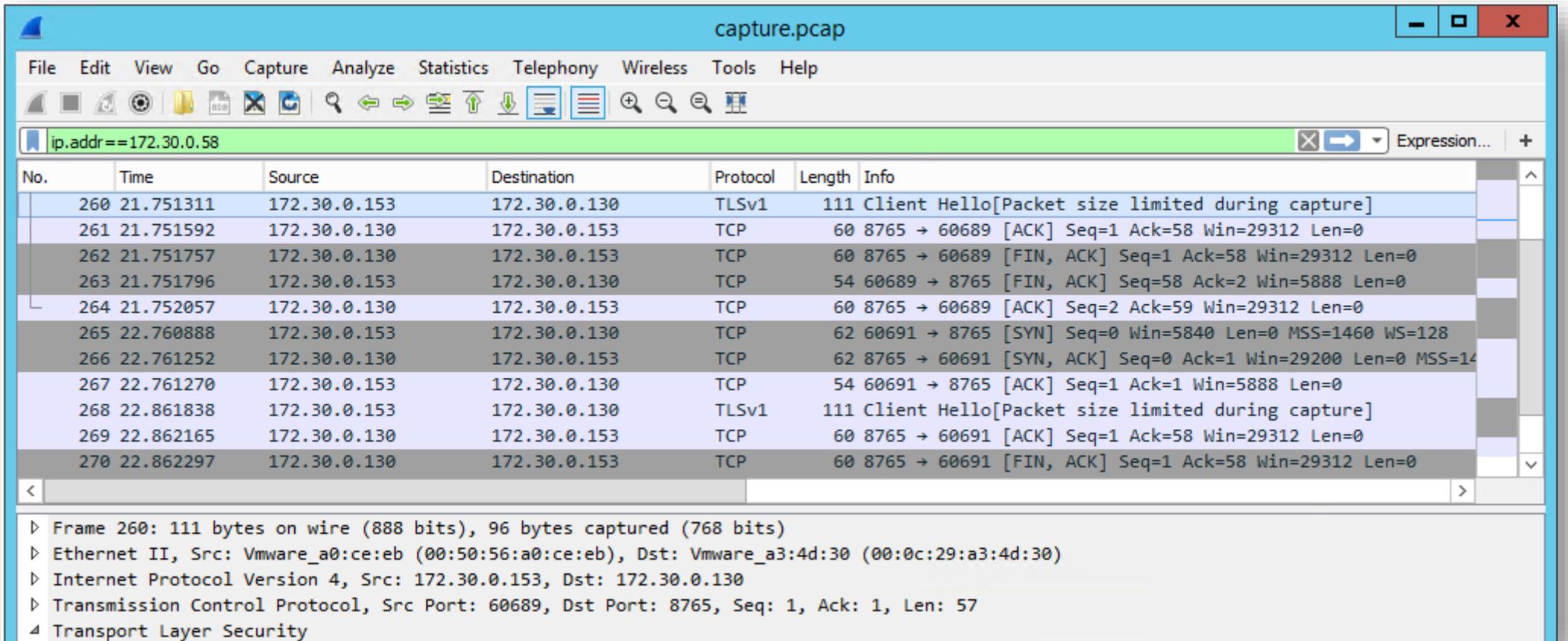
Troubleshooting using Packet Capture

Use WinSCP to copy the file to a Windows PC...



Troubleshooting using Packet Capture

Open the capture with Wireshark



The screenshot shows the Wireshark interface with a packet capture named 'capture.pcap'. The filter bar is set to 'ip.addr==172.30.0.58'. The packet list pane shows the following data:

No.	Time	Source	Destination	Protocol	Length	Info
260	21.751311	172.30.0.153	172.30.0.130	TLSv1	111	Client Hello[Packet size limited during capture]
261	21.751592	172.30.0.130	172.30.0.153	TCP	60	8765 → 60689 [ACK] Seq=1 Ack=58 Win=29312 Len=0
262	21.751757	172.30.0.130	172.30.0.153	TCP	60	8765 → 60689 [FIN, ACK] Seq=1 Ack=58 Win=29312 Len=0
263	21.751796	172.30.0.153	172.30.0.130	TCP	54	60689 → 8765 [FIN, ACK] Seq=58 Ack=2 Win=5888 Len=0
264	21.752057	172.30.0.130	172.30.0.153	TCP	60	8765 → 60689 [ACK] Seq=2 Ack=59 Win=29312 Len=0
265	22.760888	172.30.0.153	172.30.0.130	TCP	62	60691 → 8765 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 WS=128
266	22.761252	172.30.0.130	172.30.0.153	TCP	62	8765 → 60691 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
267	22.761270	172.30.0.153	172.30.0.130	TCP	54	60691 → 8765 [ACK] Seq=1 Ack=1 Win=5888 Len=0
268	22.861838	172.30.0.153	172.30.0.130	TLSv1	111	Client Hello[Packet size limited during capture]
269	22.862165	172.30.0.130	172.30.0.153	TCP	60	8765 → 60691 [ACK] Seq=1 Ack=58 Win=29312 Len=0
270	22.862297	172.30.0.130	172.30.0.153	TCP	60	8765 → 60691 [FIN, ACK] Seq=1 Ack=58 Win=29312 Len=0

The packet details pane for the selected packet (No. 260) shows the following structure:

- Frame 260: 111 bytes on wire (888 bits), 96 bytes captured (768 bits)
- Ethernet II, Src: Vmware_a0:ce:eb (00:50:56:a0:ce:eb), Dst: Vmware_a3:4d:30 (00:0c:29:a3:4d:30)
- Internet Protocol Version 4, Src: 172.30.0.153, Dst: 172.30.0.130
- Transmission Control Protocol, Src Port: 60689, Dst Port: 8765, Seq: 1, Ack: 1, Len: 57
- Transport Layer Security

Troubleshooting using Packet Capture

Client hello looks good -

The screenshot shows the Wireshark interface with a packet capture filter set to `ip.addr==172.30.0.58`. The packet list pane shows several packets, with packet 260 selected. The packet details pane for packet 260 is expanded to show the Transport Layer Security (TLS) section. The TLS section is expanded to show the TLSv1 Record Layer: Handshake Protocol: Client Hello. The content type is Handshake (22), the version is TLS 1.0 (0x0301), and the length is 52. The handshake protocol section is also expanded, showing the Client Hello handshake. A note indicates that the packet size was limited during capture, resulting in a truncated TLS record.

No.	Time	Source
260	21.751311	172.30.0.153
261	21.751592	172.30.0.130
262	21.751757	172.30.0.130
263	21.751796	172.30.0.153
264	21.752057	172.30.0.130
265	22.760888	172.30.0.153
266	22.761252	172.30.0.130
267	22.761270	172.30.0.153
268	22.861838	172.30.0.153
269	22.862165	172.30.0.130
270	22.862297	172.30.0.130

Transport Layer Security

- TLSv1 Record Layer: Handshake Protocol: Client Hello
 - Content Type: Handshake (22)
 - Version: TLS 1.0 (0x0301)
 - Length: 52
 - Handshake Protocol: Client Hello

[Packet size limited during capture: TLS truncated]

Frame 260: 111 bytes on wire (888 bits)
Ethernet II, Src: Vmware_a0:ce:eb (00:50:56:a0:ce:eb), Dst: Vmware_a3:4d:30 (00:0c:29:a3:4d:30)
Internet Protocol Version 4, Src: 172.30.0.153, Dst: 172.30.0.130
Transmission Control Protocol, Src Port: 60689, Dst Port: 8765, Seq: 1, Ack: 1, Len: 57
Transport Layer Security

Example – A Mismatch

But there is no server hello sent back... What could be the problem?

capture.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr==172.30.0.58

No.	Time	Source	Destination	Protocol	Length	Info
260	21.751311	172.30.0.153	172.30.0.130	TLSv1	111	Client Hello[Packet size limited during capture]
261	21.751320	172.30.0.130	172.30.0.153	TCP	60	8765 → 60689 [ACK] Seq=1 Ack=58 Win=29312 Len=0
262	21.751329	172.30.0.153	172.30.0.130	TCP	60	8765 → 60689 [FIN, ACK] Seq=1 Ack=58 Win=29312 Len=0
266	22.761252	172.30.0.130	172.30.0.153	TCP	62	8765 → 60691 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
267	22.761270	172.30.0.153	172.30.0.130	TCP	54	60691 → 8765 [ACK] Seq=1 Ack=1 Win=5888 Len=0
268	22.861838	172.30.0.153	172.30.0.130	TLSv1	111	Client Hello[Packet size limited during capture]
269	22.862165	172.30.0.130	172.30.0.153	TCP	60	8765 → 60691 [ACK] Seq=1 Ack=58 Win=29312 Len=0
270	22.862297	172.30.0.130	172.30.0.153	TCP	60	8765 → 60691 [FIN, ACK] Seq=1 Ack=58 Win=29312 Len=0

> Frame 260: 111 bytes on wire (888 bits), 96 bytes captured (768 bits)
> Ethernet II, Src: Vmware_a0:ce:eb (00:50:56:a0:ce:eb), Dst: Vmware_a3:4d:30 (00:0c:29:a3:4d:30)
> Internet Protocol Version 4, Src: 172.30.0.153, Dst: 172.30.0.130
> Transmission Control Protocol, Src Port: 60689, Dst Port: 8765, Seq: 1, Ack: 1, Len: 57
↳ Transport Layer Security

Example – A Mismatch

AES is sending hello using TLS1.0 – CM only accepts TLS 1.2

```
└─ Transport Layer Security
  └─ TLSv1 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 52
    └─ Handshake Protocol
      [Packet size limited display]
```

AVAYA

Help Log Off

Administration

Administration / Server (Maintenance)

Restarts

System Logs

Ping

Traceroute

Netstat

Server

Status Summary

Process Status

Shutdown Server

Server Date/Time

Software Versions

Minimum TLS Versions

Connection Type

System Management Interface (SMI) pages:

CM Duplication Link:

Filesync connections:

CM signaling connections:



How to Modify a Certificate

Through your troubleshooting process you have determined that there is an invalid variable in your certificate causing the failure, let's say the SAN is missing the fqdn, and the correct solution for you is to add that to the SAN. How can I modify a certificate?



How to Modify a Certificate

Through your troubleshooting process you have determined that there is an invalid variable in your certificate causing the failure, let's say the SAN is missing the fqdn, and the correct solution for you is to add that to the SAN. How can I modify a certificate?

Answer: You cannot modify a certificate. You have to start from scratch with a CSR, then go to the CA to have it signed, thus creating a new certificate entirely. If you manage the product in question, you are familiar with exactly how to create a CSR, and you manage the certificate authority then this process is a piece of cake. How often is that the case?



Conclusions

- Understand the basics of certs.
- All the products work a bit differently.
- Some perform more verification of certs (TLS versions / server names)
- Each product that says "Avaya" on it was designed by a different engineer at a different company at a different time.
 - There is no standardization between the systems, there is no standard repository for certificates or trust
 - The number of certificates varies from server to server.
- Your best bet - there is little documentation providing specific steps used to troubleshoot certificates.
 - Know exactly what certificates our system uses, when they use them, and what they are supposed to look like.

What's the best way for you to get help with security?



Find the best partner – here at the show!

Please fill out your session survey!

Please tweet about the presentation if you liked it - @clauss



ConvergeOne

- Come ask us questions
- www.convergeone.com
- Thanks for attending!



Chris Clauss

cclauss@convergeone.com



Carmen Piunno

cpiunno@convergeone.com



Trivia Time!

Question: what failures would you see in CM in “display errors” if I deleted the identity certificate in AES? What else would stop working?



Trivia Time!

Question: what failures would you see in CM in “display errors” if I deleted the identity certificate in AES? What else would stop working?

Answer: There would be no failure. The AES certificate is not used during the link setup because the AES is the client.

What else would stop working? Anything connecting to the AES using secure TSAPI or secure DMCC would definitely stop working. As the AES is the server in those circumstances.



Trivia Time!

Question: We've established that in a sip trunk between CM and Session Manager both servers can be a client or a server depending on which direction the call is going. Therefore both servers need an identity certificate. Do these two certificates need to be signed by the same certificate authority? Why?



Trivia Time!

Question: We've established that in a sip trunk between CM and Session Manager both servers can be a client or a server depending on which direction the call is going. Therefore both servers need an identity certificate. Do these two certificates need to be signed by the same certificate authority? Why?

Answer: They do not need to be signed by the same CA. They just need to trust each other's CA!



Certificate Filenames

- Fortunately, there are only two types of files; and even more fortunately there is only one type of file you'll ever see: Base64 Certificate. This is a certificate represented in ASCII text. It's encoded, not encrypted. You can name it .txt and view it, but it's just a random assortment of ascii text.
- In Windows if you want to be able to double click a certificate to view the actual specifics of the certificate you can just rename it .crt.
- In Linux openssl it does not matter what the file extension is.
- In your server you are trying to upload a new certificate, let's say into Avaya AVP, the software developer put in an arbitrary check that will not even let you upload a file unless it ends in .crt.
- Freely rename them for your own good.



Multiple Server Certificates on systems

The next step is to identify which service you are troubleshooting: is it the https web administrative gui, SIP instant message, SIP presence update, SIP phone call, cti-link, tsapi link, syslog, SLAMON, etc? These are all protocols that can be configured to use TLS.

Why do I need to know that? This TLS negotiation happens before the actual protocol communications, and it's identical for every service, so why does this matter which service it is?

Answer:

- 1) This can be a complicated subject. To simplify: There is not just 1 identity certificate on a server. There is not just 1 root certificate repository. These services do not all use the same cert or root repository. Can you install the same cert in all 7 services? Yes. Do people do that? Yes. Is that okay? Yes. Should that ever be assumed? No.
- 2) You need to know the TCP port being used so that manual prodding can be accomplished.