Client SSL Authentication on BIG-IP as in-depth as it can go

Rodrigo_Albuque
MVP

on 14-Feb-2020 00:56

Quick Intro

In this article, I'm going to explain how SSL client certificate authentication works on BIG-IP and explain what actually happens during client authentication as in-depth as I can, showing the TLS headers on Wireshark.

This article is about the client side of BIG-IP (Client SL profile) authenticating a client connecting to BIG-IP.

The Topology

For reference so we can follow Wireshark output:

How to Configure Client Certificate Authentication on Client SSL profile

Essentially, what we're doing here is making BIG-IP verify client's credentials before allowing the TLS handshake to proceed.

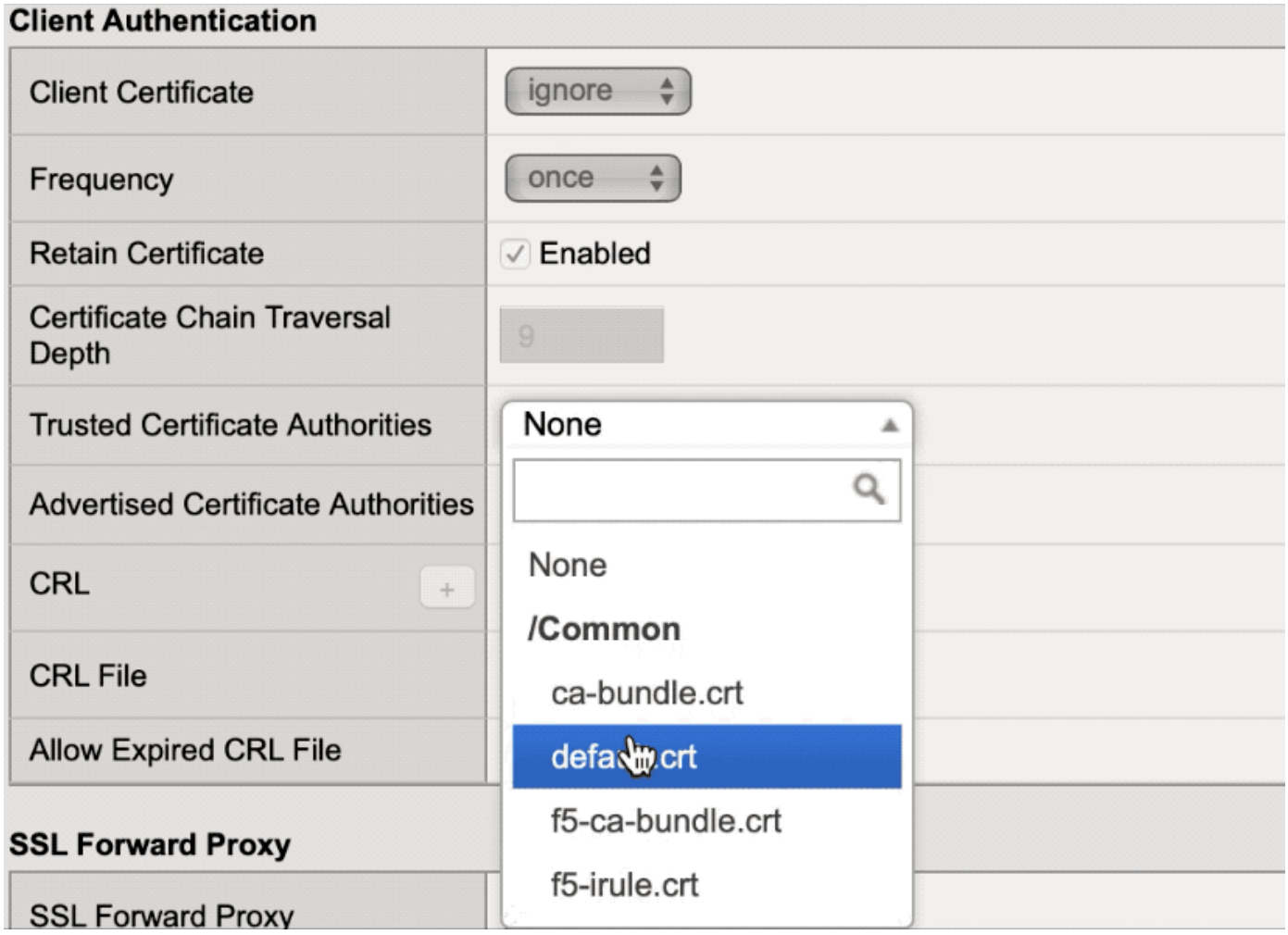
However, such credentials are in the form of a client certificate.

The way to do this is to configure BIG-IP by:

- Adding a CA file to **Trusted Certificate Authorities** (**ca-file** in **tmsh**) to validate client certificate
- Optionally adding same CA that signed client certificate to **Advertised Certificate Authorities**
- Enforcing Client Certificate validation by setting **Client Certificate** option on BIG-IP to **require**
- Optionally setting the **Frequency** of such checks if we don't want to stick to the defaults.

I'll go through each option now.

Adding CA file to Trusted Certificate Authorities



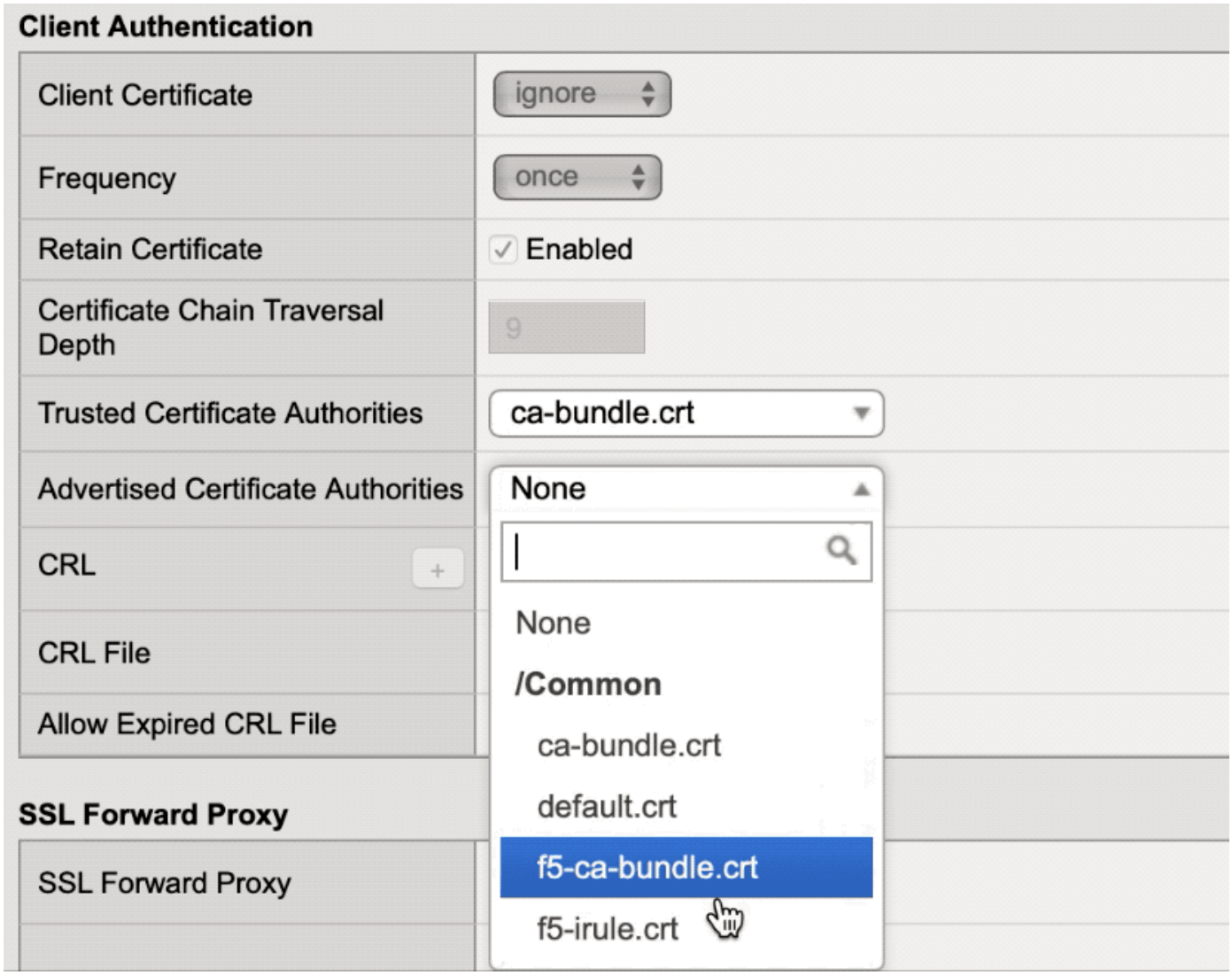
We should add to **Trusted Certificate Authorities** a single certificate file (*.crt) with one CA or concatenated file with 2 or more CAs with the purpose of validating client certificate, i.e. confirming client's identity.

Upon receiving client certificate, BIG-IP will go through this list of CAs and confirm client's identity.

It also has another purpose which is to authenticate BIG-IP to client but it's out of the scope of this article.

Optionally add CA file to Advertised Certificate Authorities

Trusted Certificate Authorities explicitly tells BIG-IP the CA or chain of CAs it will use to validate client certificate whereas **Advertised Certificate Authorities** tells client in advance what kind of CA BIG-IP trusts so that client can make the decision about which certificate to send to BIG-IP:



Why would we use Advertised Certificate Authorities?

Let's imagine a situation where Client's application has more than one Client Certificate configured.

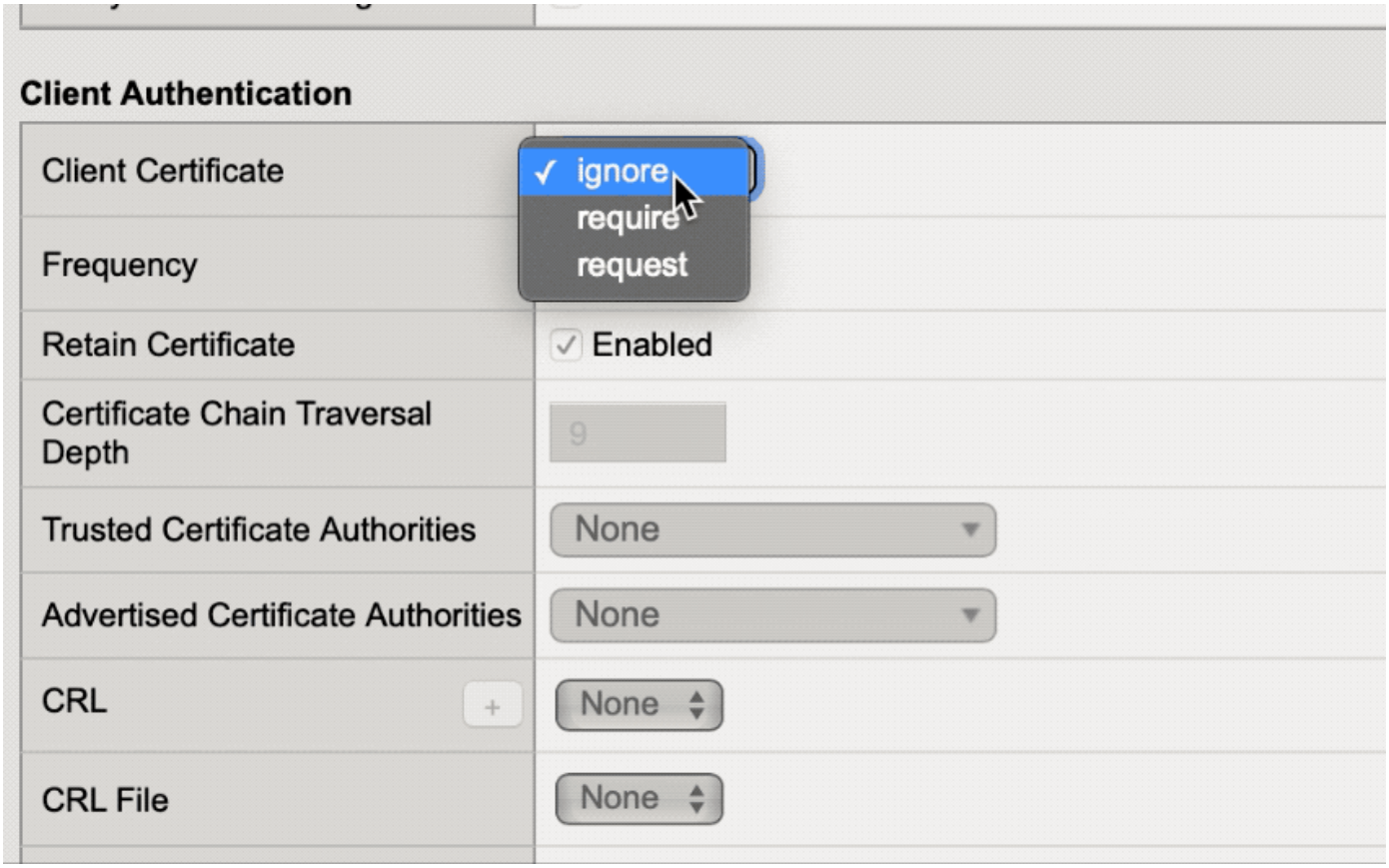
How is it going to figure out which certificate to send to BIG-IP? That's where Advertised Certificate Authorities comes to rescue us!

When we add our CA bundle to **Advertised Certificate Authorities**, we're telling BIG-IP to add it to a header field named **Distinguished Names** within **Certificate Request** message.

I dedicated the **Appendix** section to show you more in-depth how changing **Advertised Certificate Authority** affects **Certificate Request** header.

Configuring BIG-IP to enforce Client Certificate validation

To enable client certificate authentication on BIG-IP we change Local Traffic ›› Profiles : SSL : Client ›› **Client Certificate** to request:



The default is set to **ignore** where client certificate authentication is disabled.

If we truly want to enable client certificate validation we need to select **require**.

The reason why is because **request** makes BIG-IP request a client certificate from the client but BIG-IP will not perform the validation to confirm if the certificate sent is valid in this mode.

The following subsections explain each option.

Ignore

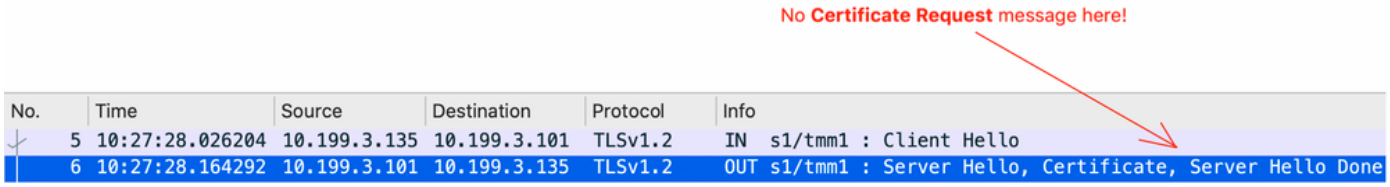
Client Certificate Authentication is disabled (the default).

BIG-IP never sends **Certificate Request** to client and therefore client does not need to send its certificate to BIG-IP.

In this case, TLS handshake proceeds successfully without any client authentication:

pcap: ssl-sample-peer-cert-mode-ignore.pcap

Wireshark filter used: frame.number == 5 or frame.number == 6

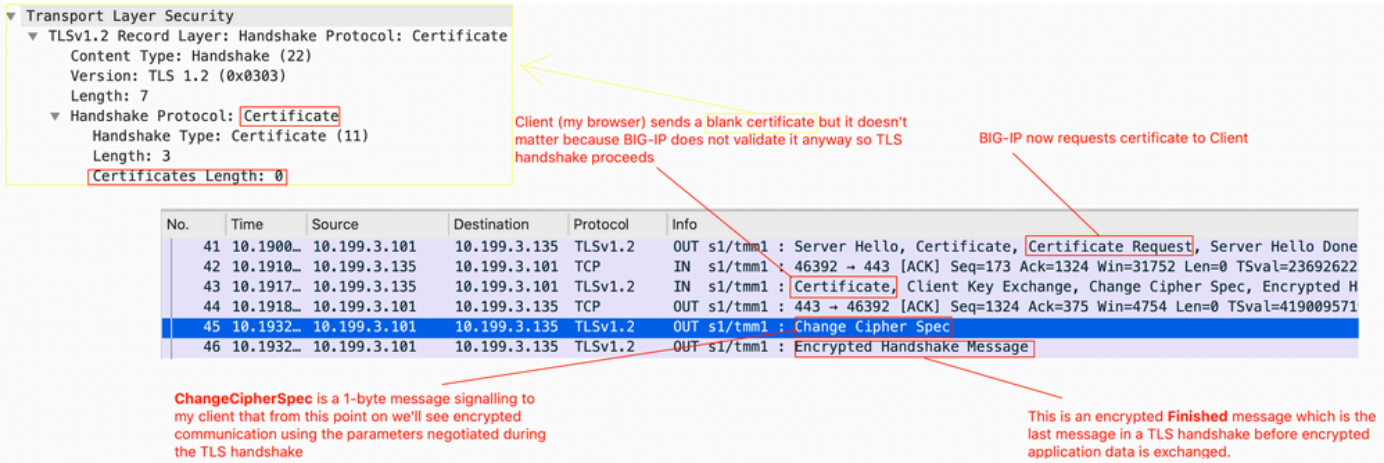


Request

BIG-IP requests Client Certificate by sending **Certificate Request** message but does not check whether client certificate is valid which is not really client authentication, is it?

This means that **ca-file** (Trusted Certificate Authorities in the GUI) will not be used to validate client certificate and we will consider any certificate sent to us to be valid.

For example, in ssl-sample-peer-cert-mode-request-with-no-client-cert-sent.pcap we now see that BIG-IP sends a **Certificate Request** message and client responds with a **Certificate** message this time:



Because I didn't add any client certificate to my browser, it sent a blank certificate to BIG-IP.

Again, BIG-IP did not perform any validation whatsoever, so TLS handshake proceeded successfully.

We can then conclude that this setting only makes BIG-IP request client certificate and that's it.

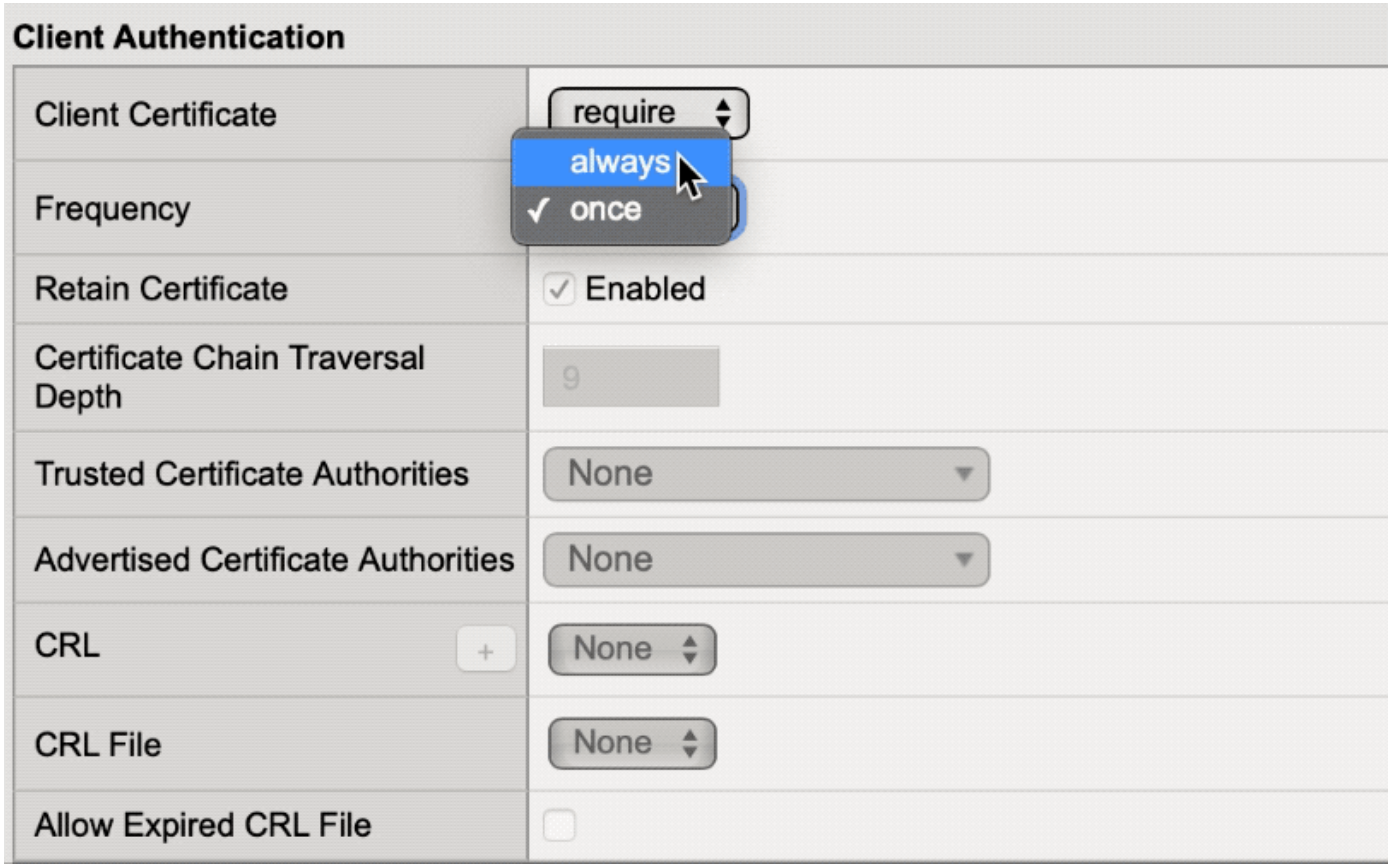
Require

It behaves just like **Request** but BIG-IP also performs client certificate validation, i.e. BIG-IP will use CA we hopefully added to **ca-file** (Trusted Certificate Authorities in the GUI) to confirm if client certificate is valid. This means that if we don't add a CA to Trusted Certificate Authorities (**ca-file** in tmsh) then validation will fail.

Setting the Frequency of Client Certificate Requests

This setting specifies the frequency BIG-IP authenticates client by enabling/disabling TLS session resumption.

It has only 2 options:



once

BIG-IP requests client certificate during first handshake and no longer re-authenticates client as long as TLS session is reused and valid.

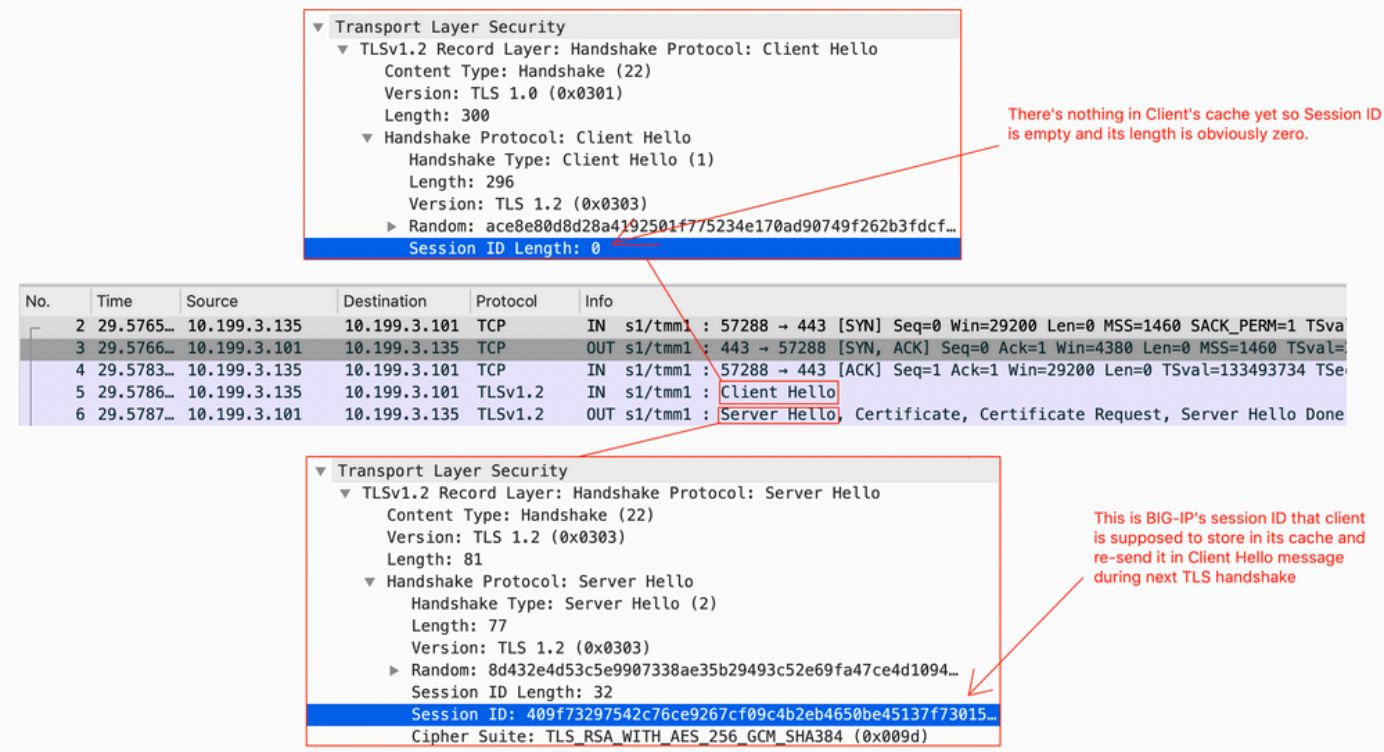
The way BIG-IP does it is by using Session Resumption/Reuse.

During first TLS handshake from client, BIG-IP sends a **Session ID** to Client within **Server Hello** header and in subsequent TLS connections, assuming **session ID** is still in BIG-IP's cache and client re-sends it back to BIG-IP, then session will be resumed every time client tries to establish a TLS session (respecting cache timeout).

First time client sends **Client Hello** with blank **session ID** as it's cache is empty and then it is assigned a **Session ID** by BIG-IP (**409f...**😞

pcap: ssl-sample-clientcert-auth-once-enabled.pcap

Wireshark filter used: filter: !ip.addr == 172.16.199.254 and frame.number > 1 and frame.number < 7



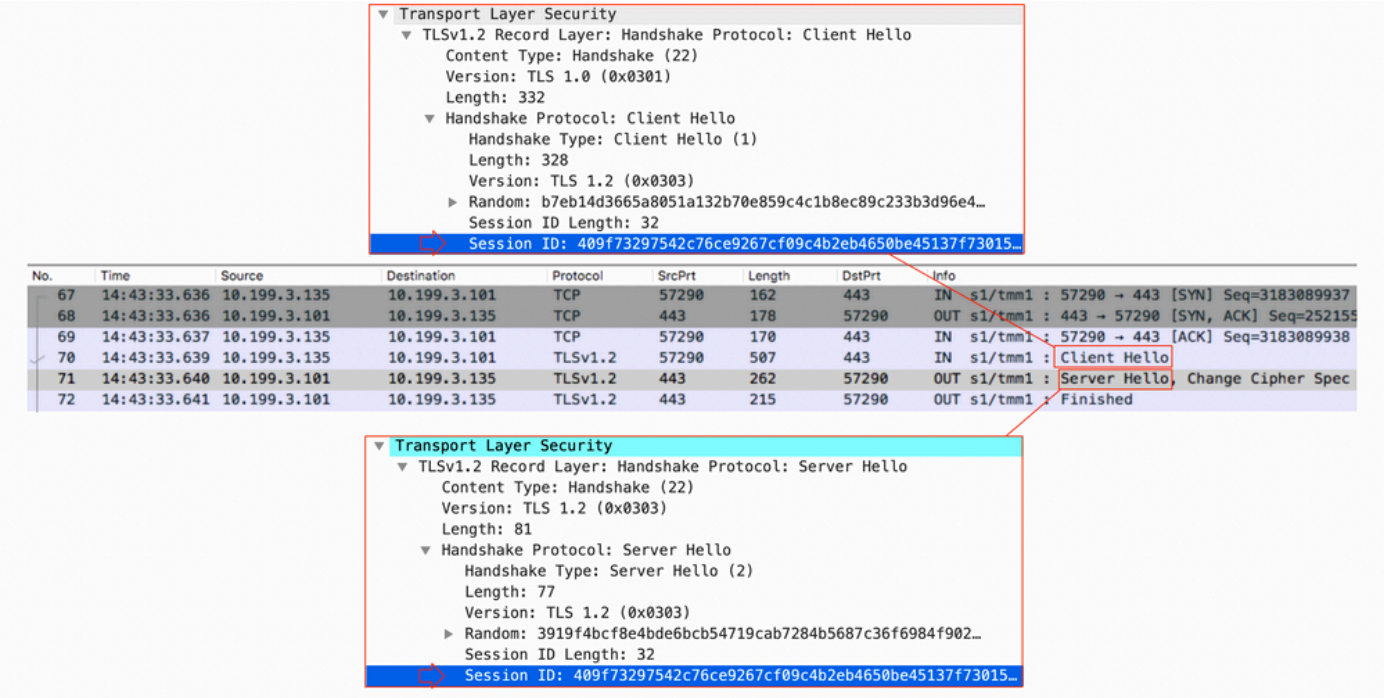
Certificate Request confirms BIG-IP is trying to authenticate client.

Notice **Session ID** BIG-IP sent to client is **409f7...**

Then when Client tries to go through another TLS handshake and sends above session ID in **Client Hello** (packet #70 below).

BIG-IP then confirms session is being resumed by sending same **session ID** in **Server Hello** back to client:

Wireshark filter used: !ip.addr == 172.16.199.254 and frame.number > 66 and frame.number < 73



This resumed TLS handshake just means we will not go through full handshake and will no longer need to exchange keys, select ciphers or re-authenticate client as we're reusing those already negotiated in the full TLS handshake where we first received **Session ID 409f...**

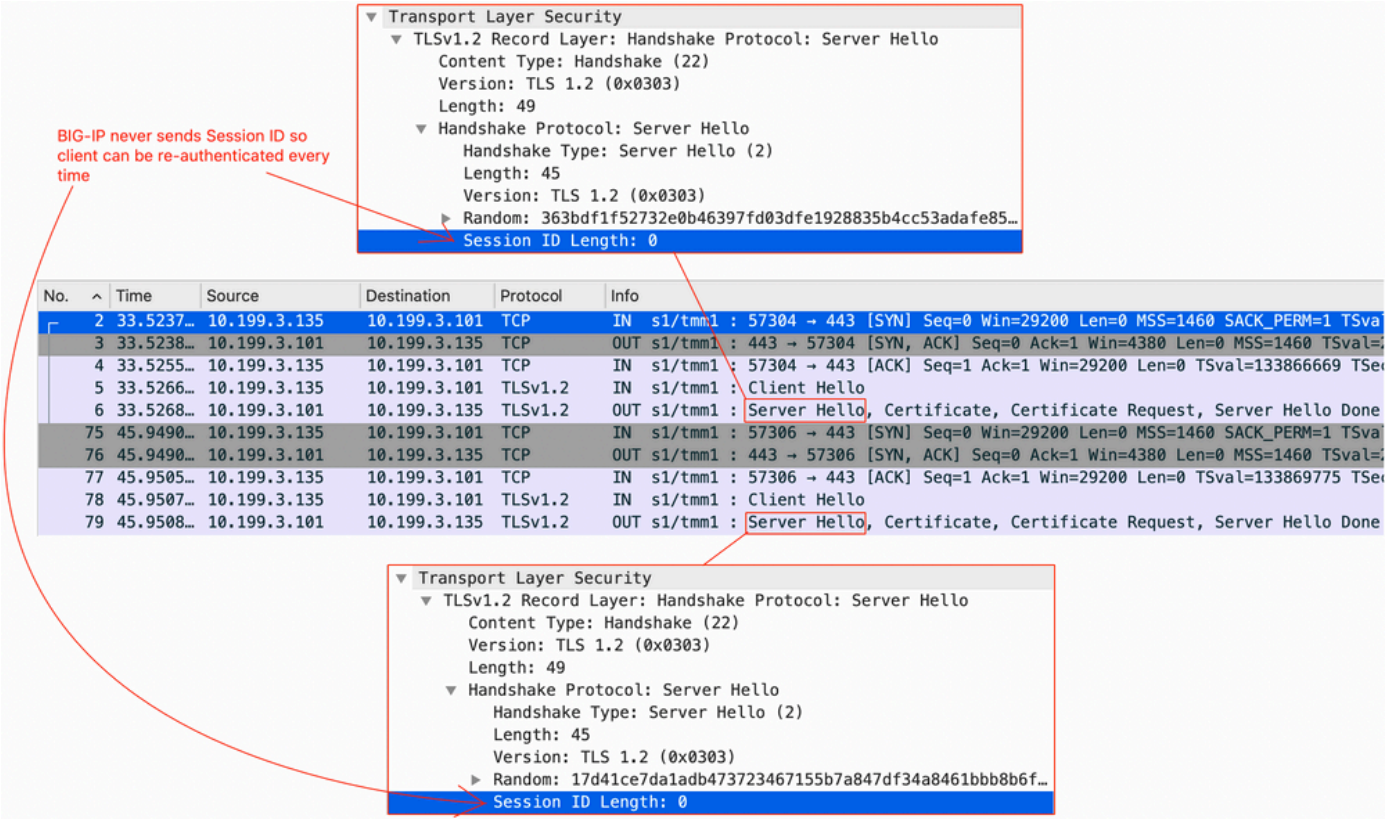
always

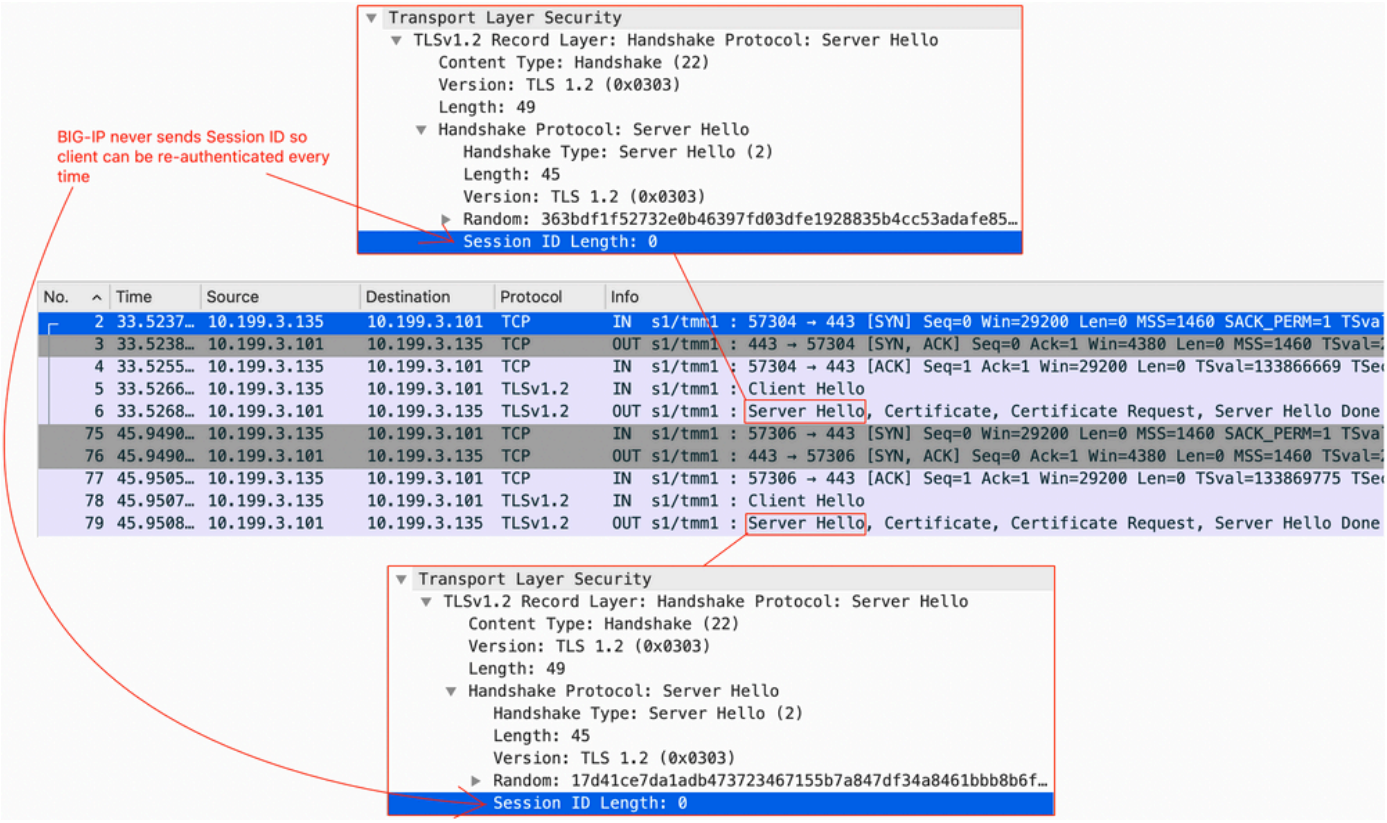
BIG-IP requests client certificate, i.e. re-authenticates client at every handshake.

On BIG-IP, this is accomplished by disabling session reuse which makes BIG-IP not to send **Session ID** back to Client in the beginning and forcing a full TLS handshake every time.

pcap: ssl-sample-clientcert-auth-always-enabled.pcap

Wireshark filter used: !ip.addr == 172.16.199.254 and ((frame.number > 1 and frame.number < 7) or (frame.number > 74 and frame.number < 80))




BIG-IP never sends Session ID so client can be re-authenticated every time
Transport Layer Security
TLSv1.2 Record Layer: Handshake Protocol: Server Hello
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 49
Handshake Protocol: Server Hello
Handshake Type: Server Hello (2)
Length: 45
Version: TLS 1.2 (0x0303)
Random: 363bdf1f52732e0b46397fd03dfe1928835b4cc53adafe85…
Session ID Length: 0
No. Time Source Destination Protocol Info
2 33.5237… 10.199.3.135 10.199.3.101 TCP IN s1/tmm1 : 57304 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSva
3 33.5238… 10.199.3.101 10.199.3.135 TCP OUT s1/tmm1 : 443 → 57304 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460 TSval=
4 33.5255… 10.199.3.135 10.199.3.101 TCP IN s1/tmm1 : 57304 → 443 [ACK] Seq=1 Ack=1 Win=29200 Len=0 TSval=133866669 TSe
5 33.5266… 10.199.3.135 10.199.3.101 TLSv1.2 IN s1/tmm1 : Client Hello
6 33.5268… 10.199.3.101 10.199.3.135 TLSv1.2 OUT s1/tmm1 : Server Hello, Certificate, Certificate Request, Server Hello Done
75 45.9490… 10.199.3.135 10.199.3.101 TCP IN s1/tmm1 : 57306 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSva
76 45.9490… 10.199.3.101 10.199.3.135 TCP OUT s1/tmm1 : 443 → 57306 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460 TSval=
77 45.9505… 10.199.3.135 10.199.3.101 TCP IN s1/tmm1 : 57306 → 443 [ACK] Seq=1 Ack=1 Win=29200 Len=0 TSval=133869775 TSe
78 45.9507… 10.199.3.135 10.199.3.101 TLSv1.2 IN s1/tmm1 : Client Hello
79 45.9508… 10.199.3.101 10.199.3.135 TLSv1.2 OUT s1/tmm1 : Server Hello, Certificate, Certificate Request, Server Hello Done
Transport Layer Security
TLSv1.2 Record Layer: Handshake Protocol: Server Hello
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 49
Handshake Protocol: Server Hello
Handshake Type: Server Hello (2)
Length: 45
Version: TLS 1.2 (0x0303)
Random: 17d41ce7da1adb473723467155b7a847df34a8461bbb8b6f…
Session ID Length: 0

Appendix: Understanding how Advertised Certificate Authority field affects Certificate Request header

For this test, I've got the following:

- **myCAbundle.crt**: concatenation of **root_ca.crt** and **ltm2.crt** (signed by **root_ca.crt**)
- **client_cert.crt**: added to Firefox and signed by **ltm2.crt**

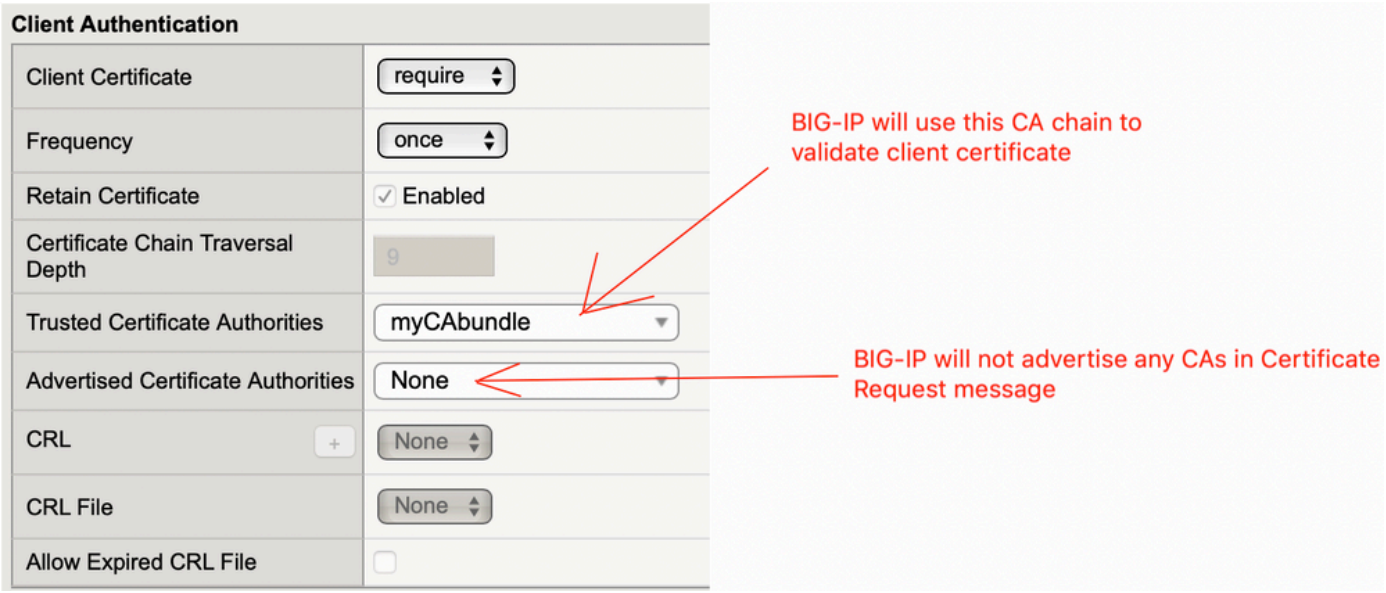
I've also added **myCAbundle.crt** to **Trusted Certificate Authorities** so BIG-IP is able to verify that **client_cert.crt** is valid.

For each test, I will use change **Advertised Certificate Authorities** so we can see what happens.

We'll go through 3 tests here:

- Setting Advertised Certificate Authority to None
- Setting Advertised Certificate Authority to a certificate that didn't sign client cert
- Setting Advertised Certificate Authority to a bundle that signed client cert

Setting Advertised Certificate Authority to None



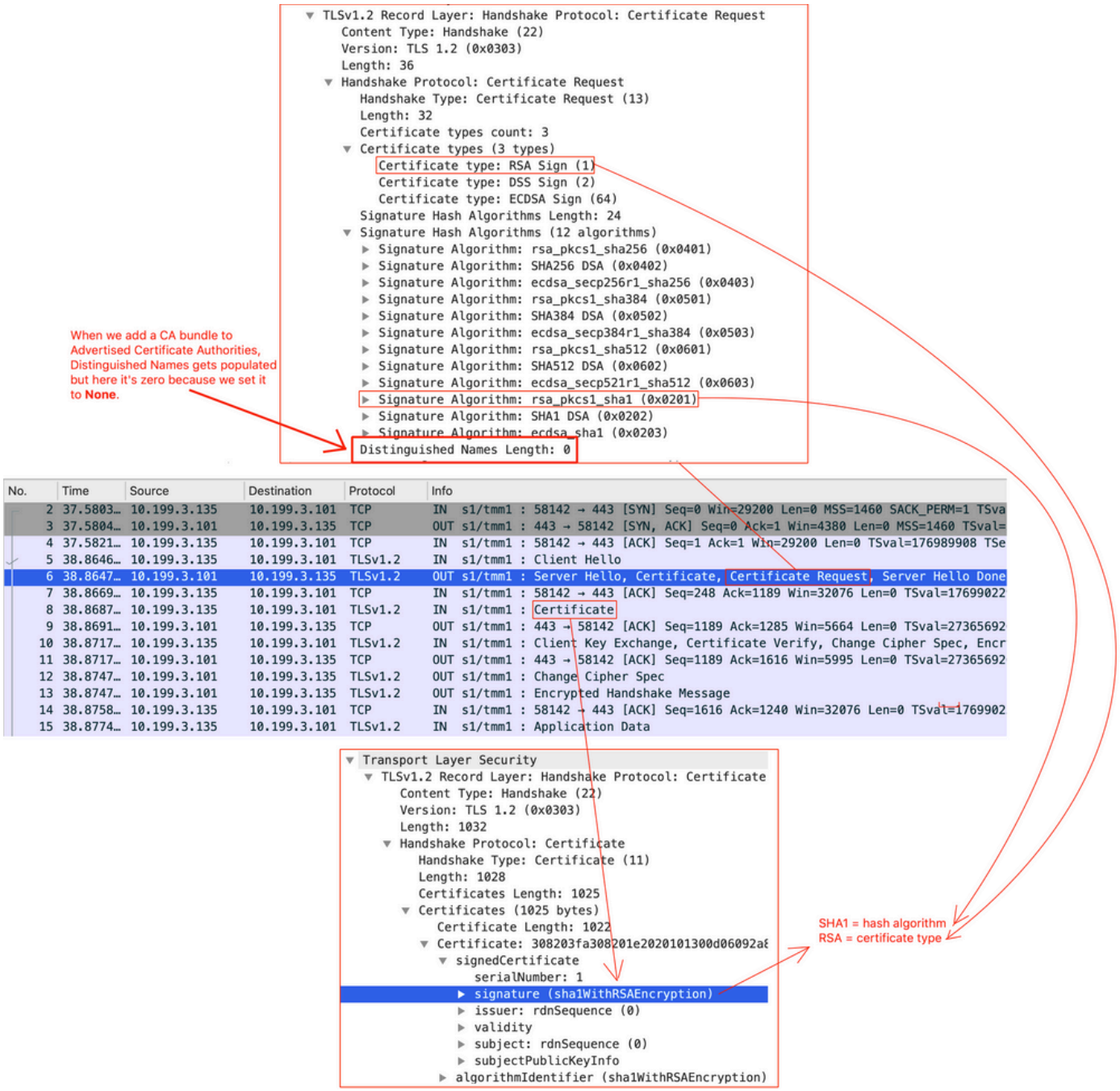
Note that even though no CA was advertised in **Certificate Request** message, BIG-IP still advertises **Certificate types** and **Signature Hash Algorithms** so that client knows in advance what kind of certificate (RSA, DSS or ECDSA) and hash algorithms BIG-IP supports in advance.

If client certificate had not been signed using any of the certificate types and hashing algorithms listed then handshake would have failed.

However, in this case validation is successful as we can see on **frame #8** that client certificate is RSA type and hashed with SHA1:

pcap: ssl-sample-advcert-none.pcap

filter used: !ip.addr == 172.16.199.254 and frame.number > 1 and frame.number < 16



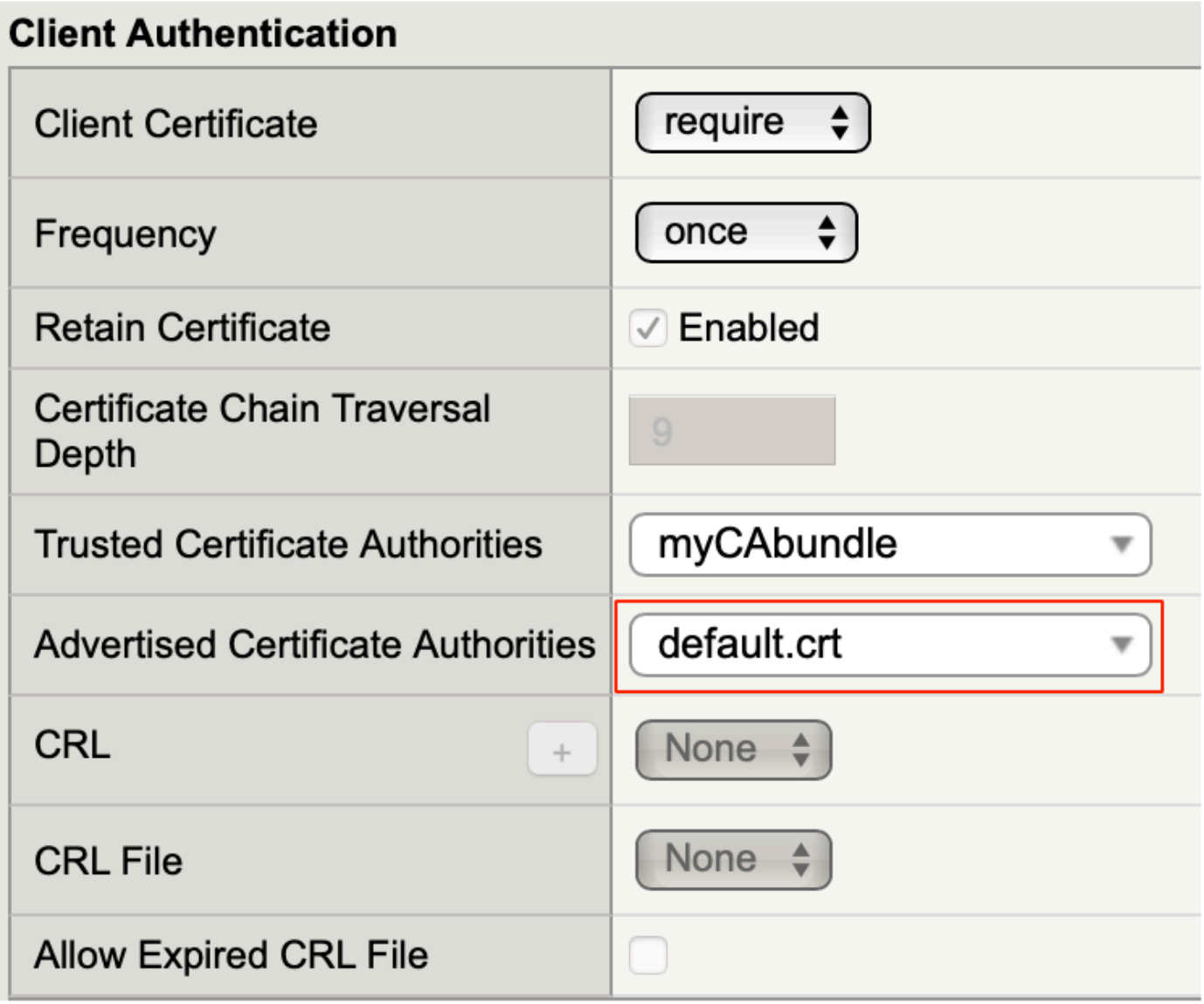
It's worth noting that Distinguished Names is NOT populated and has length of zero because we didn't attach a bundle to **Advertised Certificate Authorities**. In this case, it worked fine because my client browser had only one certificate attached, so it shouldn't cause a problem anyway.

Setting Advertised Certificate Authority to a certificate that didn't sign client cert

pcap: ssl-sample-advcert-default-firefox.pcap

Wireshark filter used: None

I've set Advertised Certificate Authority to **default.crt** as this is NOT the CA that signed client's certificate:



The difference here when compared to **None** is that now we can see that **Distinguished Names** is now populated with the certificate I added (**default.crt**😞

However, even though I added the correct certificate to my Firefox browser, it sent a blank certificate instead. Why?

That's because BIG-IP signalled in **Distinguished Names** that **default.crt** is the CA that signed the certificate BIG-IP is looking for and as Firefox doesn't have any certificate signed by **default.crt**, it just sent a blank certificate back to BIG-IP.

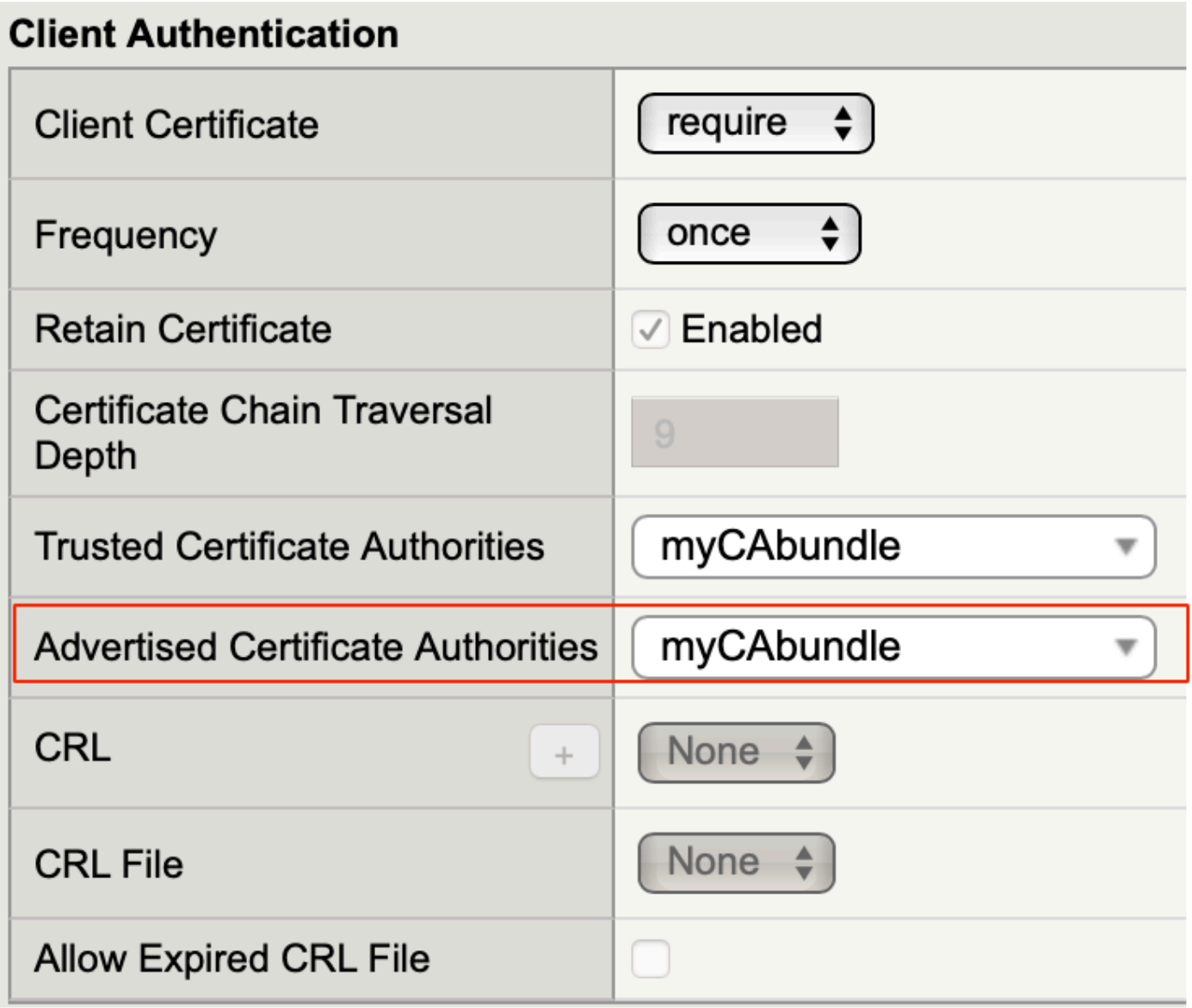
Also, because BIG-IP is now performing proper validation, i.e. comparing whatever client certificate is sent to it with the CA list added to **Trusted Certificate Authorities**, it knows a blank certificate is not valid and terminates the TLS handshake with a **Fatal Alert**.

Setting Advertised Certificate Authority to a bundle that signed client cert

pcap: ssl-sample-advcert-ltm2chainedwithrootca.pcap

filter used: !ip.addr == 172.16.199.254 and frame.number > 1

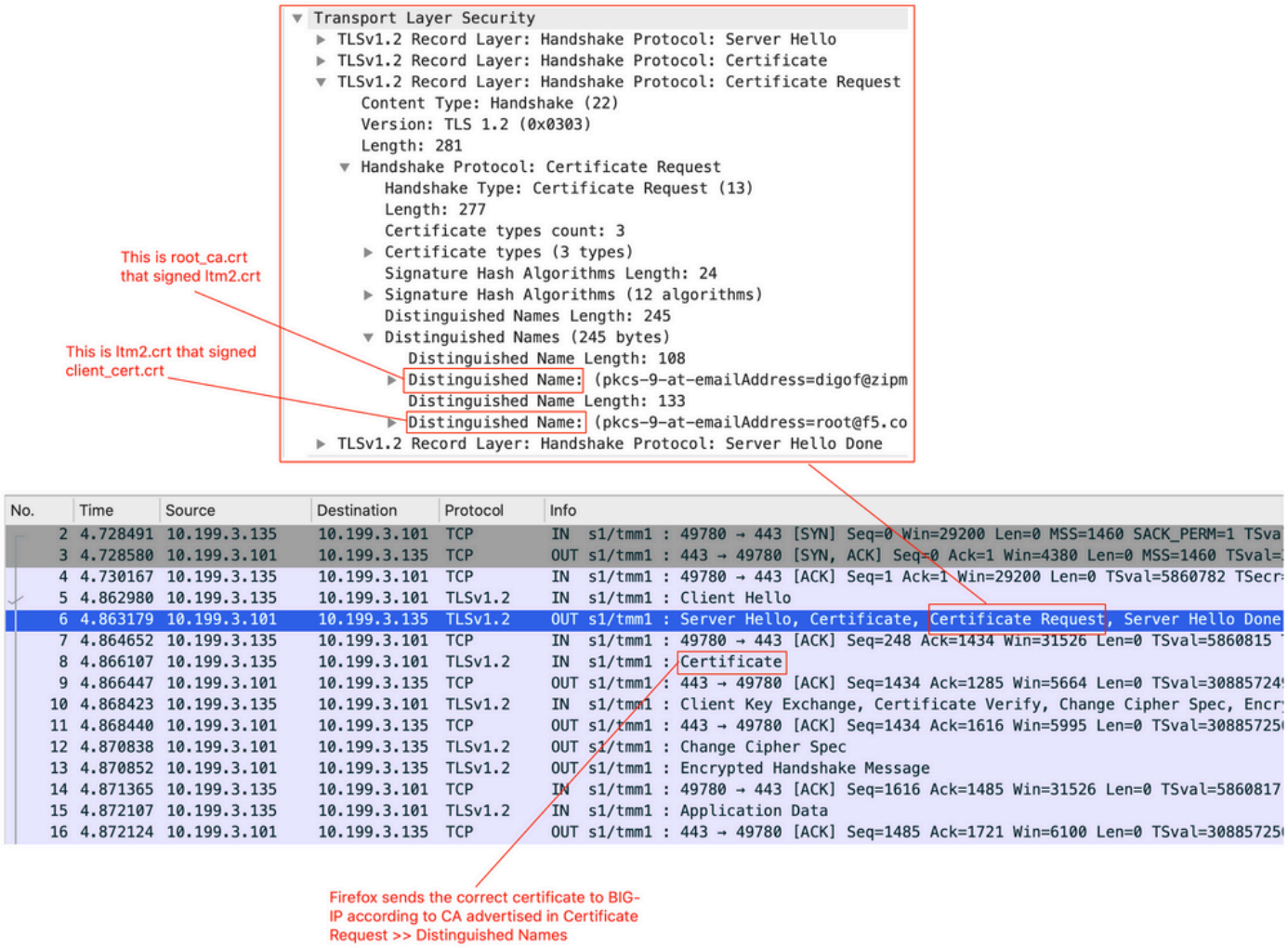
Now I've set Advertised Certificate Authority to the correct bundle that signed my client certificate:



And indeed handshake succeeds because:

- BIG-IP advertises myCAbundle.crt in **Certificate Request >> Distinguished Names** header, as per **Advertised Certificate Authority** configuration
- By reading **Distinguished Names** field, Client correctly sends the correct client certificate back to BIG-IP
- BIG-IP correctly validates client certificate using myCAbundle configured on **Trusted Certificate**

Authorities



Hope this article provides some clarification about these mysterious TLS headers.

Security

authentication BIG-IP ssl tls

6 Kudos

Comments