Understand RADIUS MTU and Fragmentation on 9800 WLC

Contents

Introduction

This document describes how to configure the MTU of the RADIUS packets the WLC sends to the RADIUS sever.

Prerequisites

Requirements

Cisco recommends that you have a basic understanding of these topics:

- 9800 Wireless LAN Controller (WLC) AAA Configuration
- Authentication, Authorization and Accounting (AAA) RADIUS concepts
- Extensible Authentication Protocol EAP
- Maximum Transmission Unit (MTU)

Components Used

- Cisco Identity Service Engineer (ISE) 3.2
- Catalyst 9800 Wireless Controller Series (Catalyst 9800-L)
- Cisco IOS® XE 17.15.2
- Windows 11 wireless client

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

For the purpose of this document, the Remote Authentication Dial-In User Service (RADIUS) server used is Cisco ISE. First, it is demonstrated how the packets would flow without any outside intervention during the extensible authentication protocol (EAP) process. Next is the configuration option to change the size of the access-request the WLC sends to any RADIUS server. This option was added in IOS-XE version 17.11.

9800 RADIUS MTU

Usually, the MTU of the RADIUS packets does not matter as they are typically small and do not hit the MTU anyways. However, when one side has to send a certificate, which are usually 2-5KB, the device needs to fragment that certificate to get it under their MTU.

When the client has to send a certificate to the RADIUS server, as is the case with EAP Transport Layer Security (EAP-TLS), it presents the WLC with a situation where the packet needs to be fragmented again due to the amount of RADIUS data that has to be sent with it. Up until 17.11 the network admin had little control over this process but now engineers are given the option to manipulate the size of the access-request sent by the WLC.

EAP-TLS Packet Flow

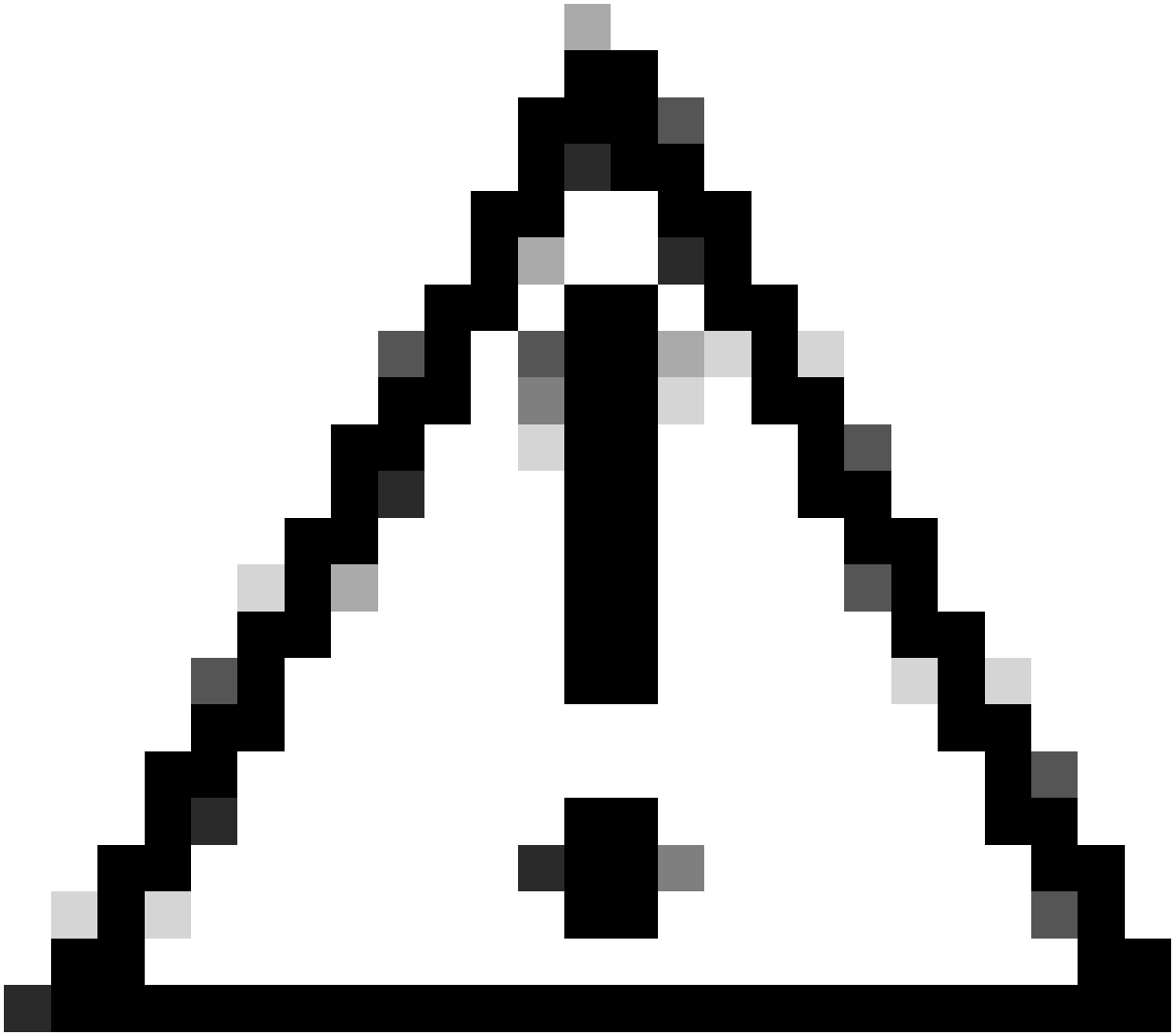
This is somewhat of a deep dive into what the packets look like and how they are treated by the wireless infrastructure. For the changes introduced in this document to be fully understood, it is important to know the packet flow during the wireless authentication process when using dot1x and more specifically EAP-TLS.

If you already have a deep understanding of how the EAP and RADIUS packet flow works with in the Cisco wireless infrastructure, then move on to the behavior change section which explains what was added in 17.11, giving the network admins more control over the RADIUS MTU. First, take a look at the EAP Identification (EAP-ID).

EAP-ID

The EAP-ID is initiated by the authenticator, in this case the WLC. This must be the first part of the EAP

process. Sometimes the wireless client sends an EAPOL-Start. This normally means that the client never received the EAP-ID request or it wants to start over.



Caution: There is a difference between the EAP-ID packet and the EAP packet ID. The EAP-ID packet is used to identify the supplicant where the EAP packet ID is a number used to track the specific packet as it moves through the network.

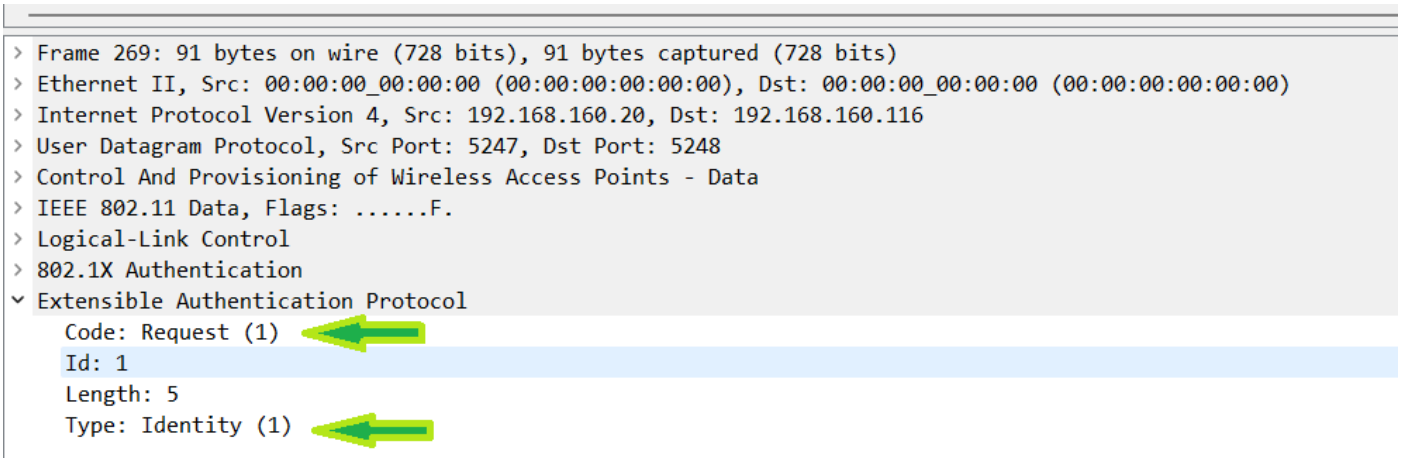
EAP-ID Request

First, the wireless client device connects to the network using the normal association process. When the Wireless Local Area Network (WLAN) is configured for dot1x, the WLC first need to know who the client is before it can request access from the RADIUS server. To find this information the WLC sends the client and EAP-ID request.

The client is expected to respond with the EAP-ID response. This gives the WLC what it needs to be able to build the access-request and send it to the ISE. The EAP-ID request is when the client would be asked to put in their username and password in a normal PEAP authentication.

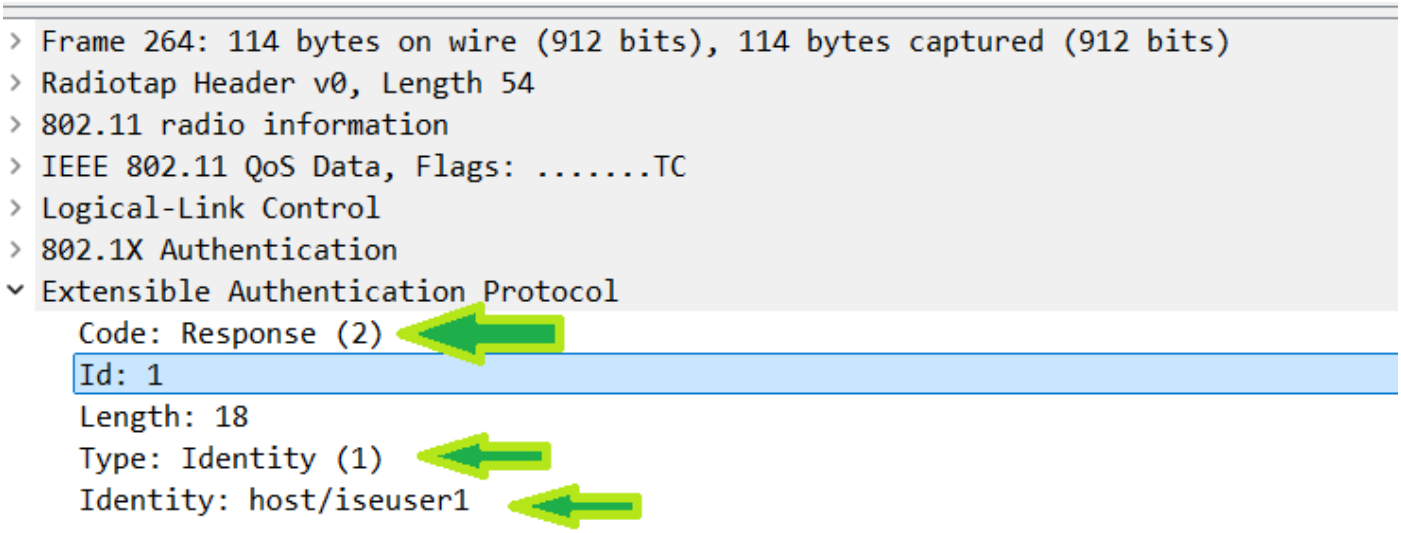
However, this discussion is around EAP-TLS so the EAP-ID response here would just have the user ID. In the demo, the user id is **iseuser1**. In this packet, you can see the EAP-ID request the WLC sends to the

wireless client asking who they are. Since this is a wireless client, the WLC encapsulates the request in CAPWAP and sends it to the Access Point (AP) to be sent over the air. In the EAP data, Code 1 signifies it is a request and type 1 signifies it is for the identity.



EAP-ID Response

Next, it is expected to see the wireless client respond with the EAP-ID response. In the EAP data the code has changed to 2 signifying that it is a response, but the type stays as a 1, still showing it is for the identity. Here, you can even see the username the client is using. One more thing to check on these packets is the ID number of the EAP packet. For the EAP-ID exchange it is always 1 but this number later changes to something else once ISE gets involved.



You can see that both packets are rather small, so the MTU has no bearing here since it is well under the 1500 bytes used in the network.

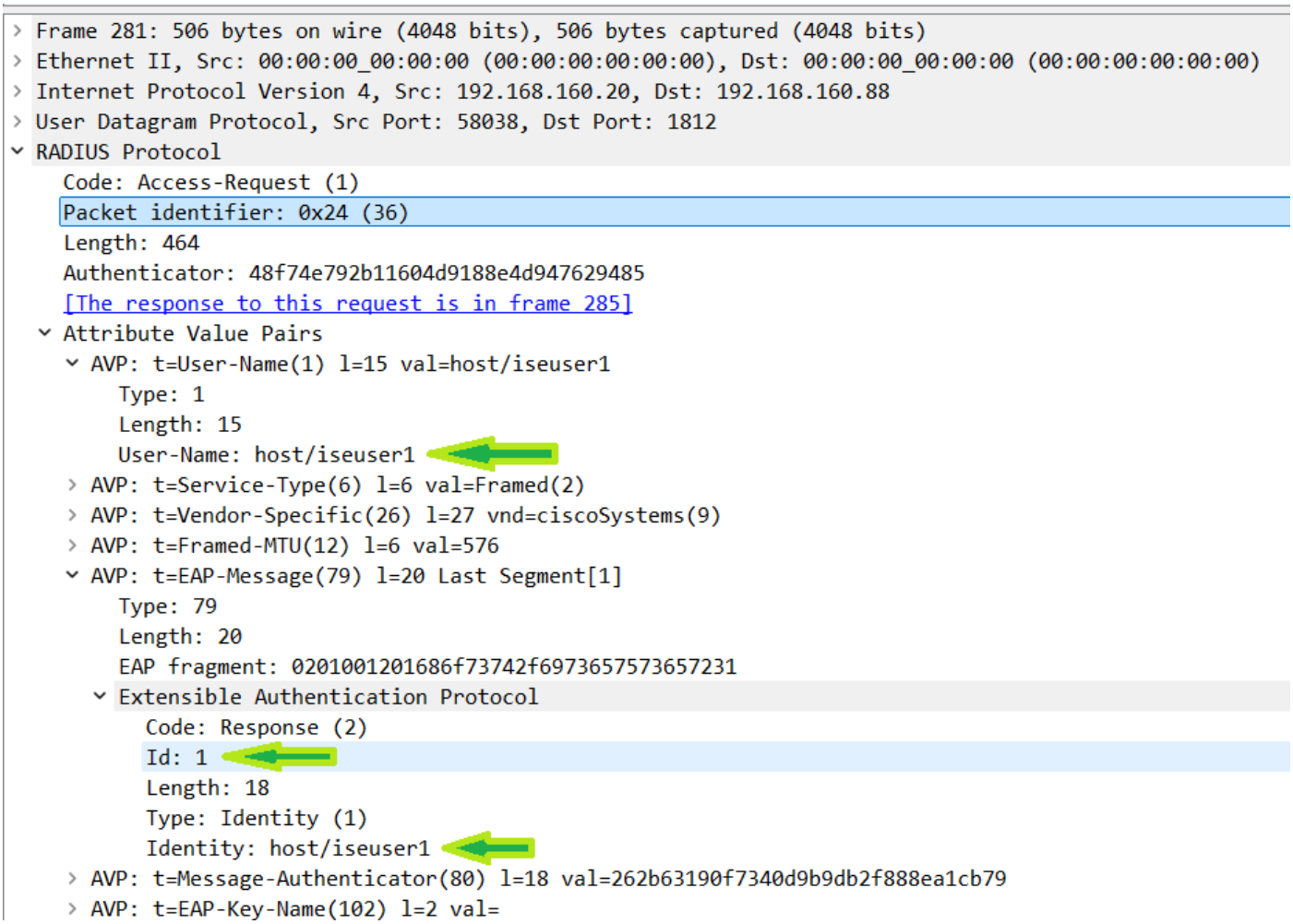
Access-Request and Access-Challenge

The communication with the client is EAP and the communication between the WLC and ISE is RADIUS. For the RADIUS communication the access-request and access-challenge packets are use. The WLC receives the EAP packet from the supplicant and forwards it to ISE using the RADIUS access-request. In a working network ISE would respond with an access-challenge.

Access-Request

Now that the WLC knows the identity of the client it needs to ask the RADIUS server if that client is allowed on the network. To do that, the WLC requests access for that client by sending the access-request packet. There are other pieces of data the WLC is going to send along with the EAP data. Collectively these are called attribute value pairs, AVPs, or AV pairs depending on who is speaking.

This document is not going to go far into the AVPs as that is outside the scope of this discussion. Here you just need to see that the username(EAP data) is included and sent to the RADIUS server, which, in this case, is ISE. Also, you can see that EAP-ID number of 1 is also sent to ISE. Remember when you looked at the EAP packet ID over air it was 1 there as well. The last important thing to note here is that since the WLC has added all these AVPs, the 114 byte packet the client sent is now turned into a 488 byte packet before being sent to ISE.



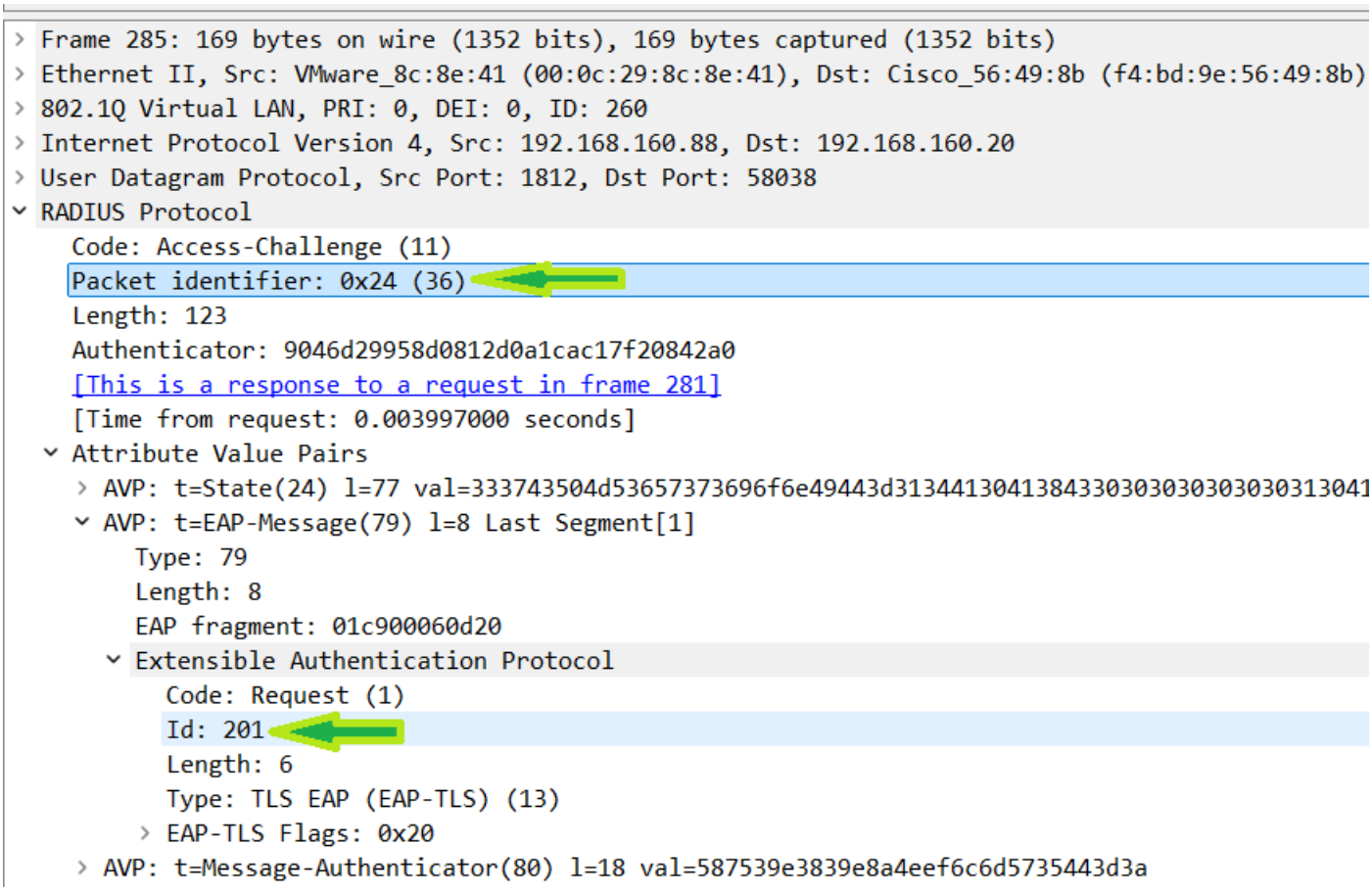
Access-Challenge

Assuming that ISE receives the access-request and decides to respond it is expected for this response to come as an access-challenge from ISE. Looking back at the access-request you would see the RADIUS packet ID of 36 before the AVPs start.

When the WLC receives the access-challenge the RADIUS ID must match that packet ID of the access-request. RADIUS packet ID is for the RADIUS communication between ISE and the WLC. You can also see in this packet that the ISE has set a new EAP ID of 201 which is used for tracking the communication between ISE and the client. At this point, the WLC is just a pass through for the communication between ISE and the client.

It is important to note all these packet IDs here so that you understand the comminication flow and how to

track these packets through the network. In a production environment there are usually multiple authentications happening at the same time. Use the **calling-station-id** command to match the packet to the MAC address of the client. Then, you can use the RADIUS packet ID and EAP packet ID to track the packet flow for this specific client. Up to this point neither side has sent any certificates, so there still has not been a need to worry about the MTU.



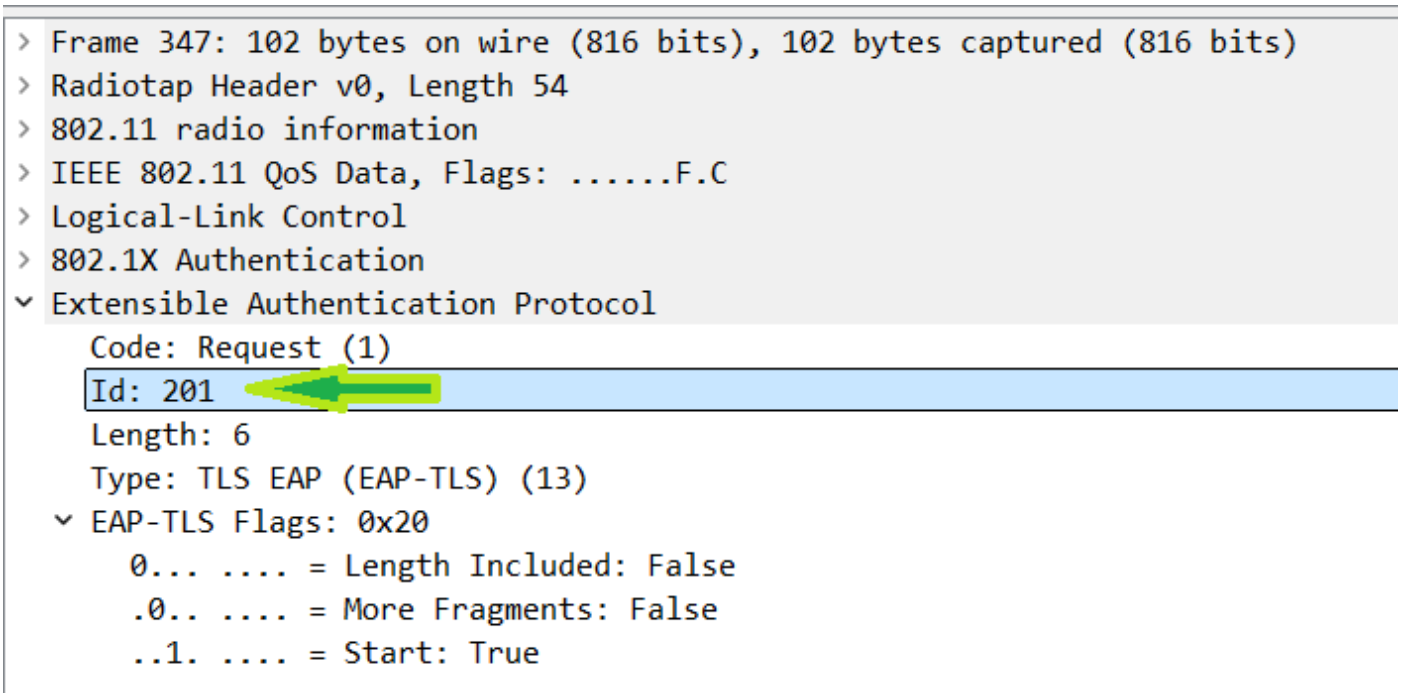
EAP Request and EAP Response

Just a reminder, the client speaks EAP not RADIUS. That said, when the WLC receives the access-challenge it has to remove the RADIUS data and pull out the EAP request so that it can be sent to the client.

EAP Request

This must look exactly as it did inside of the access-challenge when the WLC received it. However, all the RADIUS stuff has been removed and just the EAP part is sent to the client.

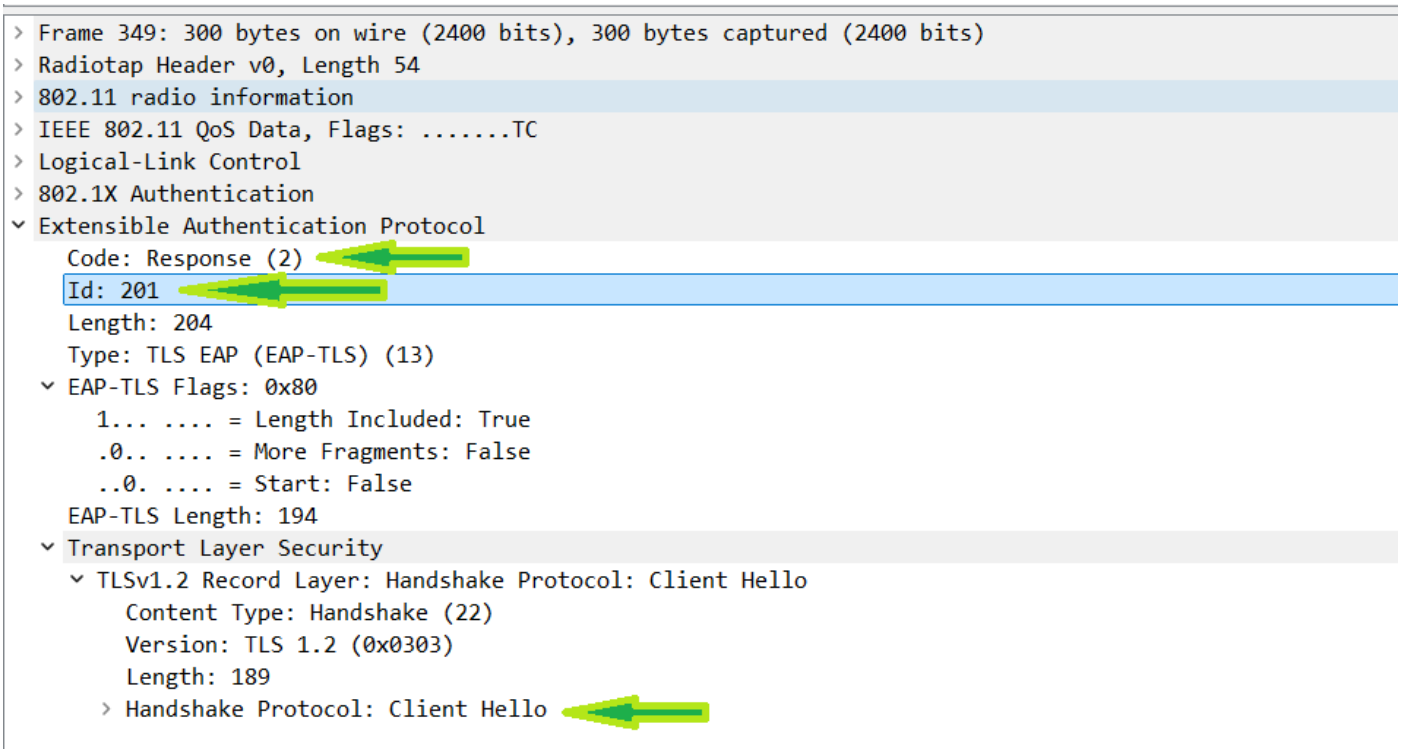
You can still see the EAP packet ID of 201 here just as it was in the access-challenge because it is the same data that the WLC received from ISE. The flow here is the same as with the EAP-ID, only now it does not come from the WLC and it is used for establishing the EAP method. This packet is still pretty small because it is just to establish the start of an EAP-TLS session.



EAP Response

When the client receives the EAP-Request, it must respond with an EAP-Response. In the EAP-Response the client starts to establish the TLS session. This looks that same as it would in any other situation TLS is use. It starts with the "client hello" message. This document is not going to dig into what goes into the client hello as it is irrelevant to this topic. What you need to notice here is just that a TLS session is being setup.

You can see the data in the packets here as you would with any other TLS setup. Just as with the EAP-ID response, this packet hits the WLC and is converted to an access-request. ISE respond with an EAP-request packaged in an access-challenge. This continues to be the flow from now on.



TLS Certificates

Here is the point where you are going to see the packet size increasing. The certificates can be rather large depending on the presence of one or more intermediate certificate authorities (CAs). If it is a self signed certificate it would obviously be smaller than a certificate with a device cert chained to 2 intermediates CAs and a root CA. Either way, you normally see the owner of the certificate start to fragment its own packets here.

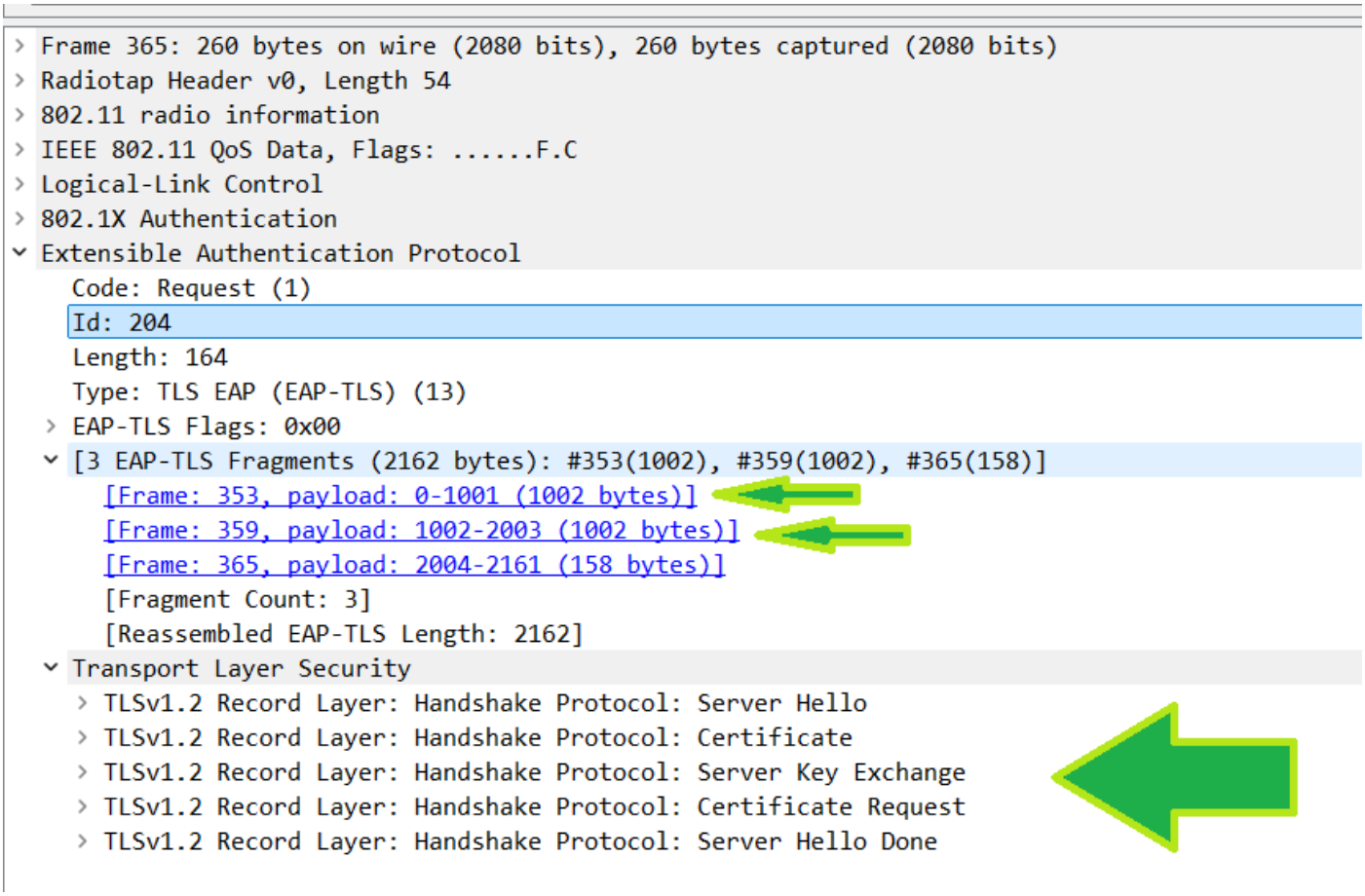
ISE Certificate

Now that ISE has received the TLS client hello, it responds with another EAP request. In this new EAP request ISE sends the "server hello" message, its certificate, the "server key exchange", the "certificate request", and the "server hello done" messages all at once. If it sent all this in one packet, the packet would be over the MTU for the network. So, ISE fragments the packet itself to get it under the MTU. With ISE, it fragments the data portion of the packet so that it is no larger than 1002 bytes in hopes of avoiding double fragmentation.

What is meant by double fragmentation? The first fragmentation is happening on ISE since the data it wants to send is too large to fit within the MTU of the network. Still, there can be other places in the network where, even though the MTU is the same, because of how the network is setup, a device possibly needs to fragment the packet in order for it to add its headers and stay under the MTU. This can be true even if the **do not fragment** bit is checked.

A good example of this is with a VPN tunnel, or any tunnel for that matter. To put data into a VPN tunnel, the VPN routers have to add their headers to the traffic. If this RADIUS traffic were fragmented at or close to the MTU, when it comes to this VPN there would be no way to keep the data under the MTU and add extra headers. This is true for CAPWAP tunnels as well which you can see a bit later.

So, to avoid these packets getting into a situation where another device can fragment it again, ISE fragments the packet at a place where this can be avoided in most networks. This means that ISE sends this data in multiple EAP Request waiting for an empty EAP response each time. The EAP ID increases with each fragment sent. From the point of view of the WLC, this would be an access-challenge and access request exchange for every fragment and the RADIUS packet ID would increase with each fragment sent.



Client Certificate

Once ISE sends all the fragments and they are reassembled by the client, the packet flow moves on to the client to send something. In TLS it is expected that the client would send its own cert at this point in order to complete authentication. This is where things become more complex. Just like ISE, the client is going to send multiple TLS parts all at once, one of these being its certificate.

Unlike what was seen with ISE, most clients send their EAP data just below the MTU. In this demo, the 802.1x data is 1492. The problem with that is that the AP needs to add the CAPWAP headers so it can be sent to the WLC.

How that be done? The AP is going to have to fragment the packet so that it can add the headers and send it to the WLC. There is no way for the AP to obtain the packet to the WLC without fragmenting it. That said, here the packet is double fragmented, first from the client, then again from the AP. However, this fragmentation is not usually a problem as it is expected with CAPWAP.

The packet over the air:

```
> Frame 367: 1588 bytes (12704 bits), 1588 bytes captured (12704 bits)
> Radiotap Header v0, Length 54
> 802.11 radio information
> IEEE 802.11 QoS Data, Flags: .......TC
> Logical-Link Control
> 802.1X Authentication
v Extensible Authentication Protocol
    Code: Response (2)
    Id: 204
    Length: 1492
    Type: TLS EAP (EAP-TLS) (13)
  v EAP-TLS Flags: 0xc0
      1... .... = Length Included: True
      .1.. .... = More Fragments: True
      ..0. .... = Start: False
    EAP-TLS Length: 4692
```

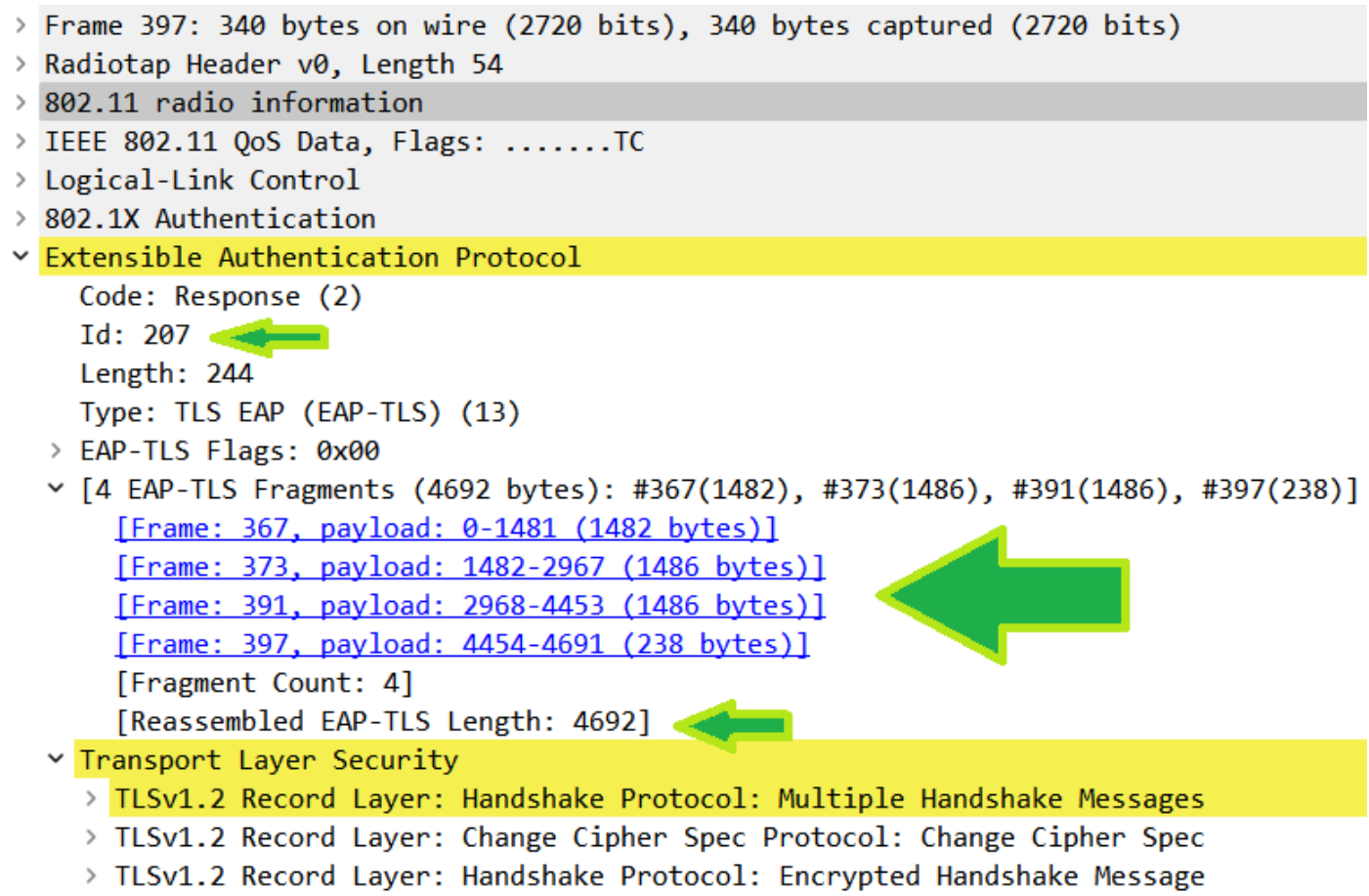
The packet fragment on the wire:

```
> Frame 56: 1482 bytes (11856 bits), 1482 bytes captured (11856 bits) on interface /tmp
> Ethernet II, Src: Cisco_b5:e6:00 (0c:75:bd:b5:e6:00), Dst: Cisco_56:49:8b (f4:bd:9e:56:49:8b)
> Internet Protocol Version 4, Src: 192.168.160.116, Dst: 192.168.160.20
> User Datagram Protocol, Src Port: 5248, Dst Port: 5247
> Control And Provisioning of Wireless Access Points - Data
  [Reassembled in: 57]
v Data (1424 bytes)
    Data: 018800000c75bdb3022038689362ec7e0c75bdb3022f00010000aaaa03000000888e0100…
    [Length: 1424]
```

The packet reassembled on the wire:

```
Wireshark · Packet 57 · FromTheWire2.pcap
> Frame 57: 156 bytes (1248 bits), 156 bytes captured (1248 bits) on interface /tmp/epc_ws/wif_to_ts_pipe, id 0
> Ethernet II, Src: Cisco_b5:e6:00 (0c:75:bd:b5:e6:00), Dst: Cisco_56:49:8b (f4:bd:9e:56:49:8b)
> Internet Protocol Version 4, Src: 192.168.160.116, Dst: 192.168.160.20
> User Datagram Protocol, Src Port: 5248, Dst Port: 5247
> Control And Provisioning of Wireless Access Points - Data
> [2 Message fragments (1530 bytes): #56(1424), #57(106)]
> IEEE 802.11 QoS Data, Flags: .......T
> Logical-Link Control
> 802.1X Authentication
v Extensible Authentication Protocol
    Code: Response (2)
    Id: 204
    Length: 1492
    Type: TLS EAP (EAP-TLS) (13)
  > EAP-TLS Flags: 0xc0
    EAP-TLS Length: 4692
```

All client fragments reassembled over the air:



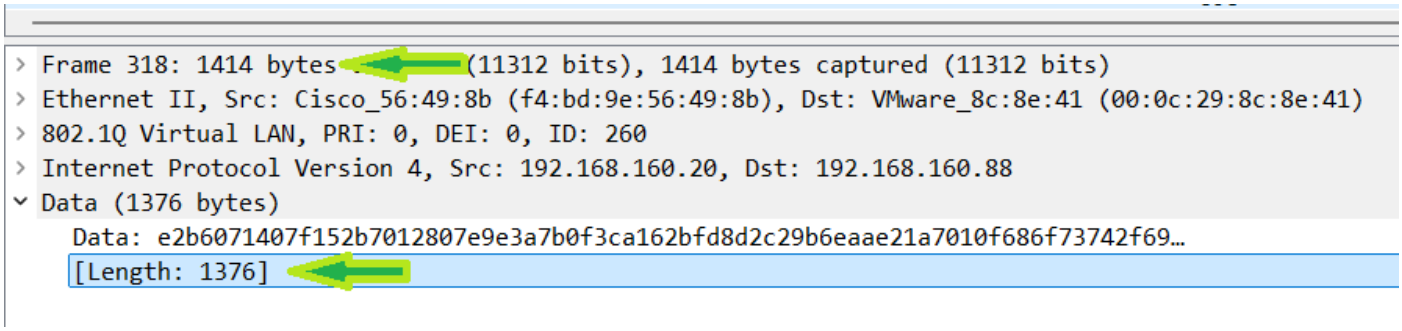
Client Cert at the WLC

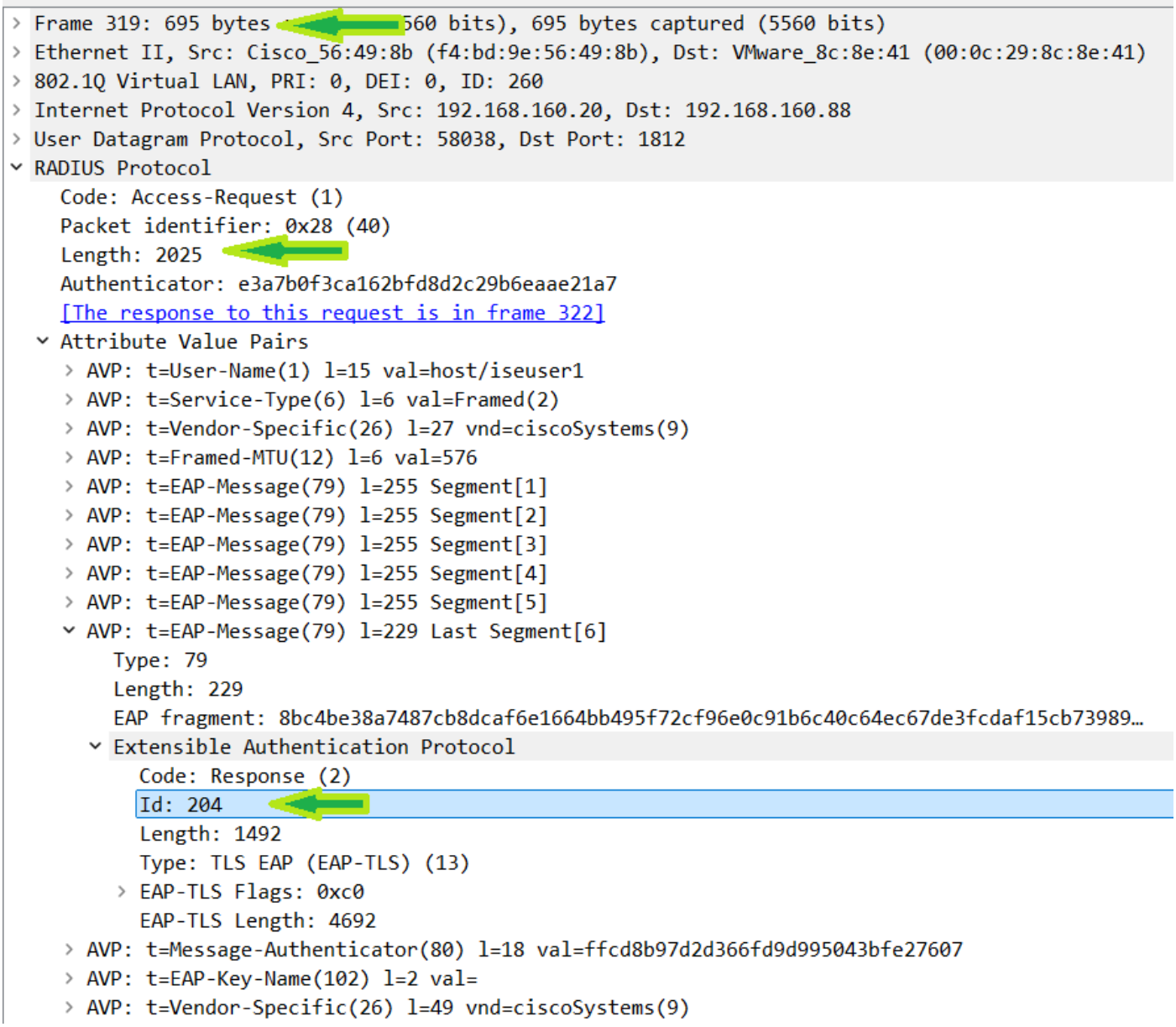
The WLC receives the two CAPWAP fragments and reassembles them to have the whole 1492 byte packet from the client, restoring the packet - but not for long. This restoration is short lived because, if you look back to how the WLC sends the access-request, you must remember that it has to add about 400 bytes worth of RADIUS AVPs to the packet before it can send the data to ISE.

For simple math, assume the WLC adds 408 bytes bringing the total packet size to 1900. This is well over the 1500 MTU so what is the WLC going to do? Fragment the packet again.

At this point, the WLC is going to fragment the packet at 1396 by default. The thought here is that same as it is with ISE. The hope is to make the packet small enough so that if it has to go through another tunnel, it wont need to be fragmented again to add the headers. However, the WLC is not as cautious as ISE so 1396 is good enough here.

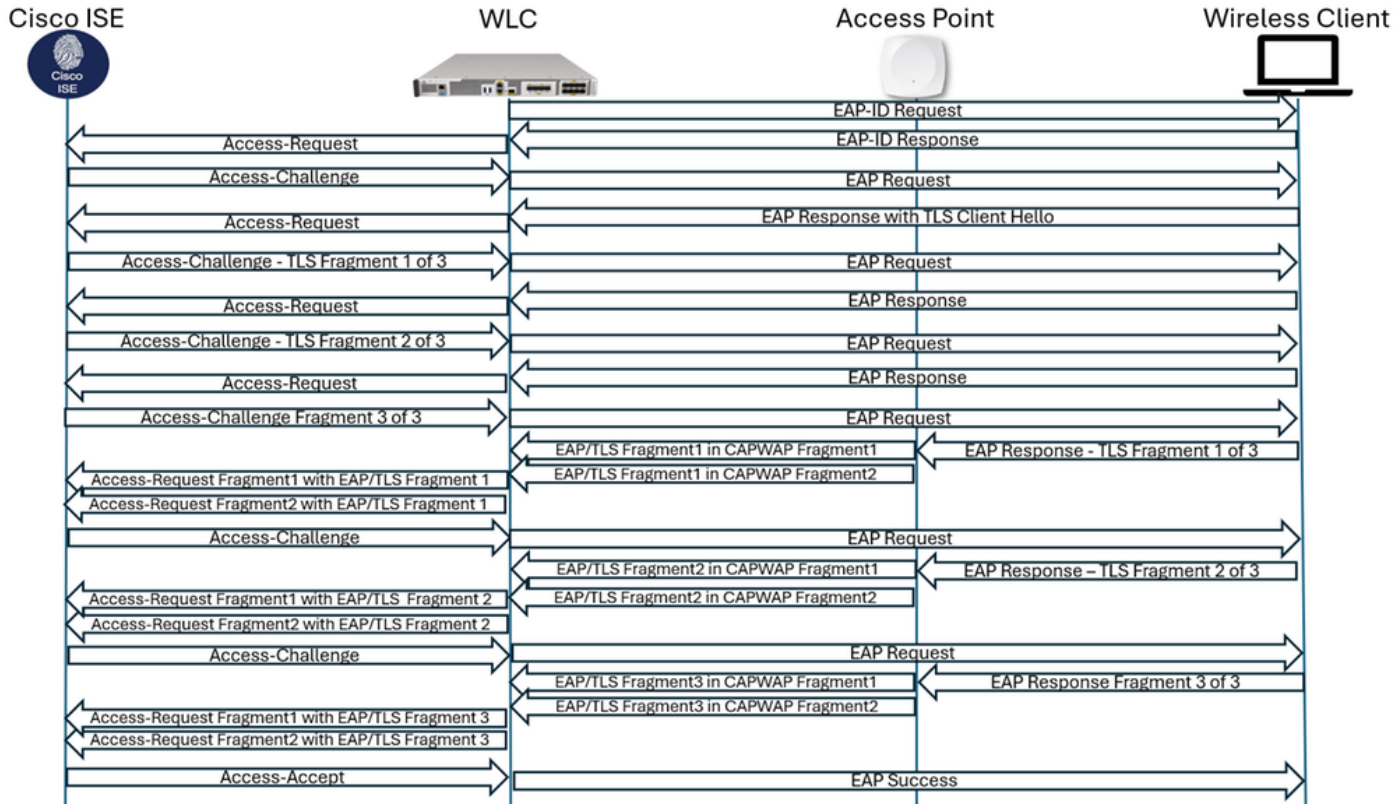
The fragmented packet leaving the WLC:





Packet Flow TL;DR

When ISE sends its certificate it fragments the TLS packets at 1002 bytes. No issues there. When the clients sends its certificate, they usually fragment close to the MTU. Since the AP has to add the CAPWAP headers to the packet it has to fragment this packet also. Once the WLc receives the fragments it has to reassemble the packet but then it has to add the RADIUS AVPs so the packet is fragmented again. The packet flow looks something like this:



RADIUS MTU Behavior Change

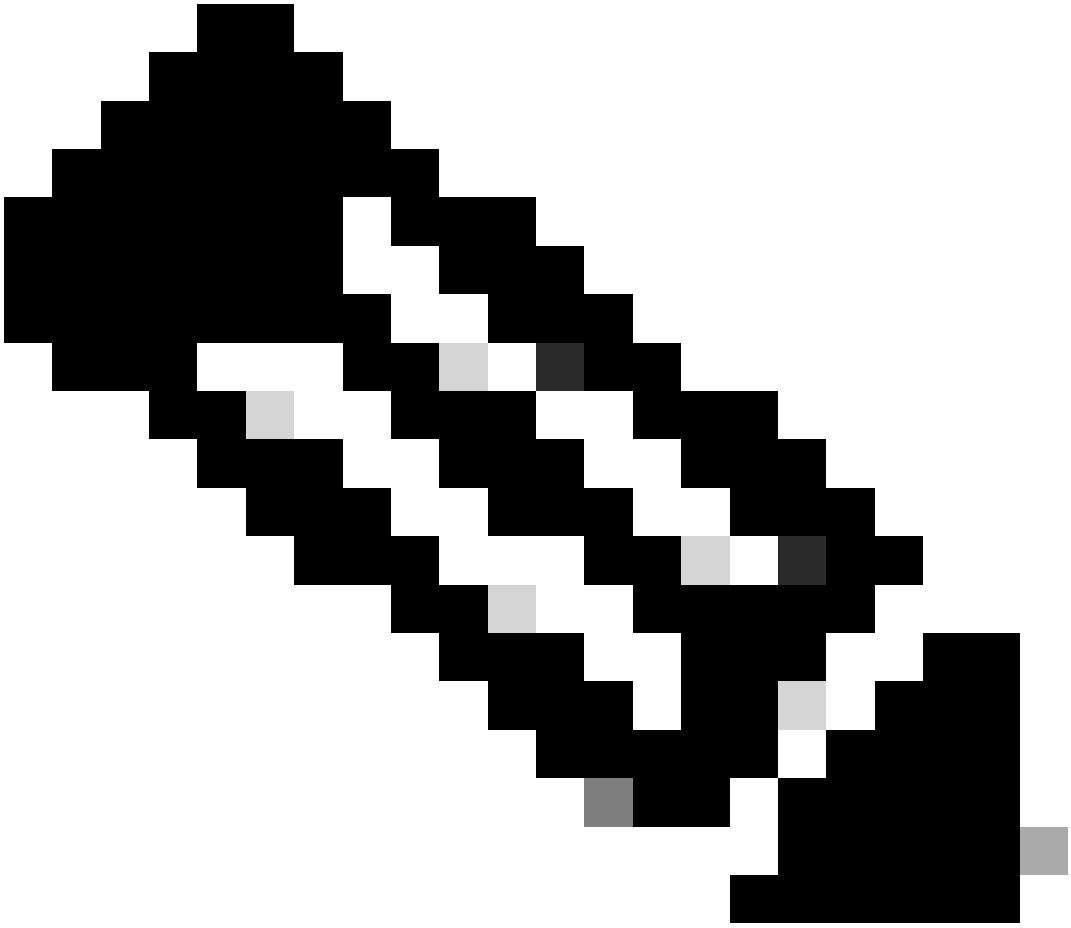
When you look at the packet flow for any wireless client data traffic, you can see that the wireless infrastructure only has influence over it at a few places. The first place is when the traffic leaves the AP and the second place is when the traffic leaves the WLC. The exception is with TCP traffic where the wireless infrastructure can adjust the client MSS. However, EAP does not fall under TCP, in fact, it is its own protocol.

When you look at the EAP and RADIUS traffic flows, you can also see that the network does in fact influence the traffic size at both the AP and WLC when the original packet size gets too close to the MTU. With a proper understanding of the role of the WLC in this exchange, you can see there is really only one place where it has influence of the size of the RADIUS packet. This would be when a EAP response comes in and you change it to a RADIUS access-request.

What Changed

If the EAP response is over the MTU, once you add the RADIUS AVPs you have to fragment it. Since you already have to fragment this packet no matter what, you can at least decide at what size you want to fragment it at. That is where the behavior change introduced in 17.11 comes in.

In the feature request tracked in Cisco bug id CSCwc81849, you want to add support for RADIUS Jumbo packets. The way this was done is that the RADIUS packet was no longer automatically fragmented at 1396. Now, if you add the **ip radius source-interface <interface X>** command the RADIUS access-request is sent at the MTU of that interface.

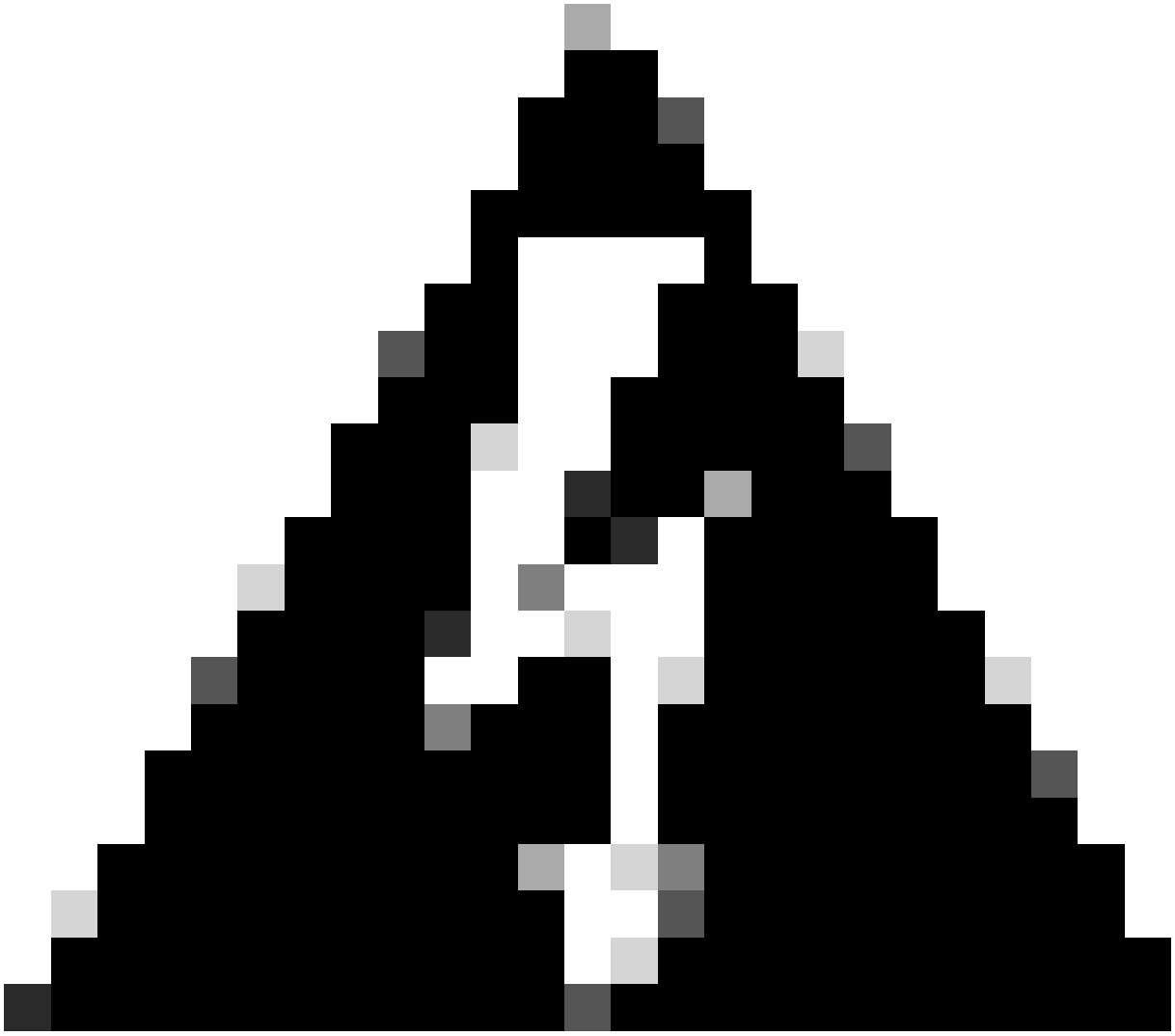


Note: If you are using Cisco Catalyst Center, when you provision AAA configs, it automatically adds the source interface to the server group. This changes the default behavior to fragment at the MTU size of the interface used in that command.

How Can This Change Be Used

Since the default MTU of all your interfaces would be 1500, that would be the new MTU to fragment at. The default interface used for all RADIUS traffic is the wireless management interface (WMI). When you look at the config of your server group, if there is no source-interface specified the WLC sends the RADIUS traffic at 1396 using the WMI. However, if you go into the server group config and tell it the source-interface is the WMI the WLC now sends the RADIUS traffic at 1500 still using the WMI.

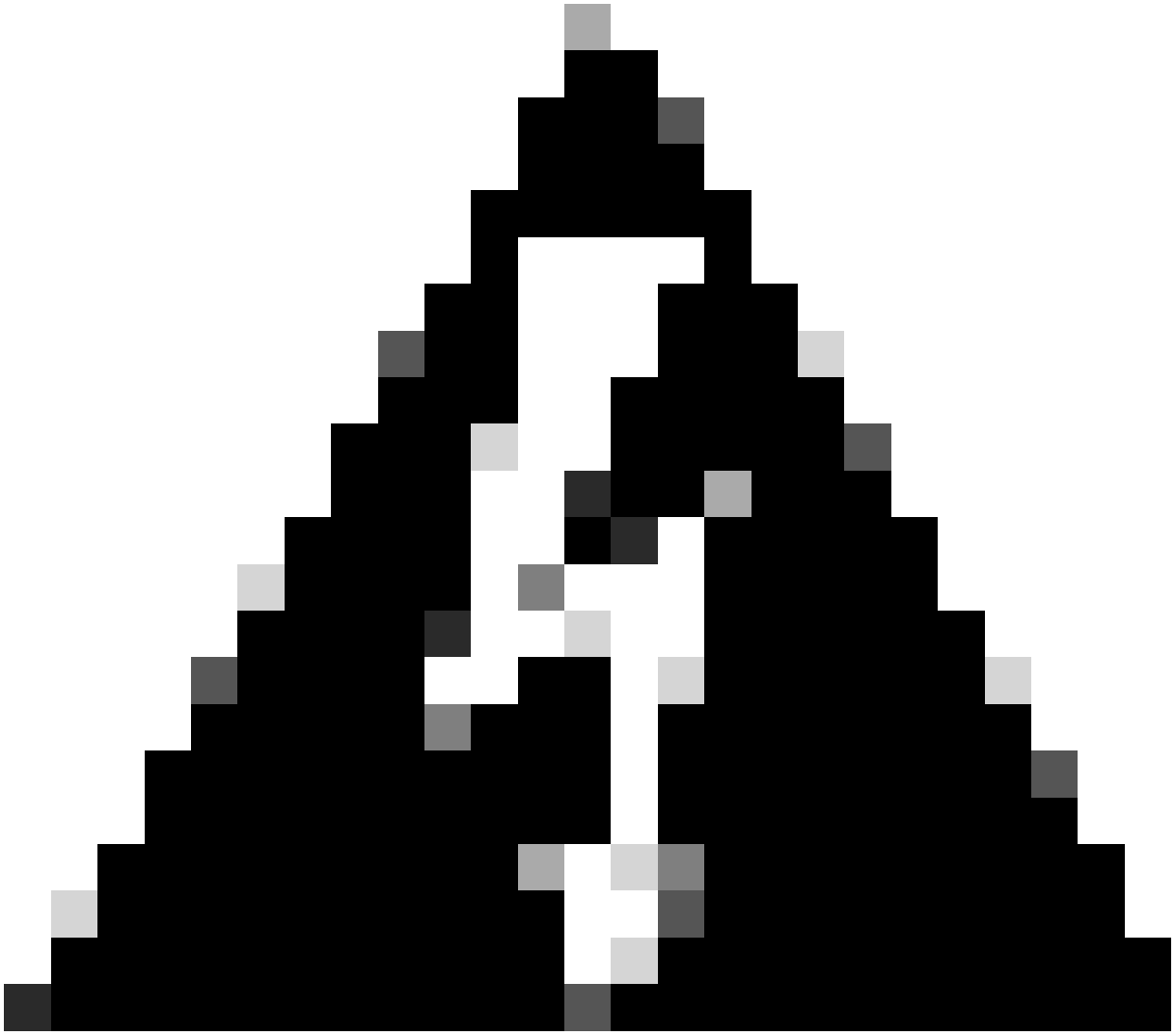
Now, assume that there is a device in the network like the VPN discussed earlier. You do not want the traffic to be double fragmented so you can change the MTU of the interface to something smaller in order to fragment the packets at a different place. You can change the MTU to something like 1200 so that the packets are fragmented at the 1200 byte mark instead of 1500.



Warning: Changing the MTU of the WMI affects all traffic going to and from the WLC management IP address.

Even though you do not want to cange the MTU of the WMI, the point of specifying a source interface is to change it from being the WMI to another interface, and use that interface for RADIUS traffic, as well as change the MTU on that interface. Since this config is done at the server group level, you can be very specific about which RADIUS traffic you want this change to be affected by.

This config is not tied to a AAA server or WLAN. It is possible to have multiple server groups with the same servers in them and only specify the source-interface on one of them if you so choose. This server group is added to a method list and then added to a WLAN. So, for example, if there is only one WLAN where you want this change to be made, even if you only have one AAA server, you can create a new server group, use the **ip radius source-interface** command that points to the interface with the MTU you want to use, add the AAA server to this new group, create a new method list using this new group, and then add that method list to the specific WLAN where you want this change to be made.



Warning: It is always suggested, when making ANY changes to a live network, it is done during a maintenance window.

The Proof is in the Packet Capture

It is commonly known that in networking, if you did not capture it, you cannot prove it. Here are couple of config examples with these changes in place to show you how this works.

Here is a WLAN config. During testing, only the server group that is used in the method list is changed.

```
9800#show run wlan
wlan TLS-Test 2 TLS-Test
 radio policy dot11 24ghz
 radio policy dot11 5ghz
 no security ft adaptive
 security dot1x authentication-list TLS-AuthC
 no shutdown
!
```

Adding The Source-Interface Command With The Default MTU

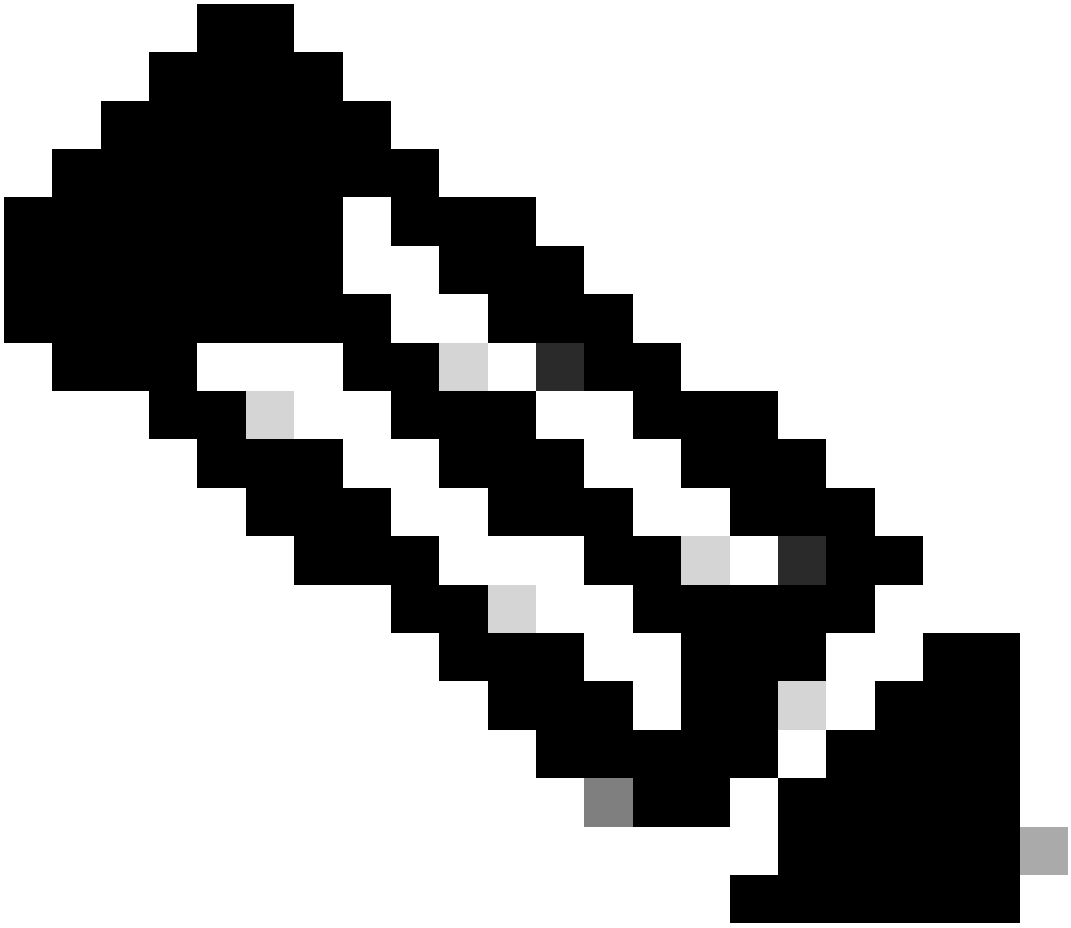
Here, it is just a normal server group pointing to the ISE server. The source interface command was added pointing to my WMI which has no MTU set. Here is what the config looks like.

```
9800#show run aaa
!
aaa authentication dot1x TLS-AuthC group NoMTU
!
!
radius server ISE
 address ipv4 192.168.160.10 auth-port 1812 acct-port 1813
 key 6 _`gINMNXObF[^bPBVNaYibbBMhNMFAbKUAAB
!
aaa group server radius NoMTU
 server name ISE
 ip radius source-interface Vlan260
 deadtime 5
!
9800#show run inter vlan 260
!
interface Vlan260
 ip address 192.168.160.20 255.255.255.0
 no ip proxy-arp
end
```

As you can see, the NoMTU server group has been added to the authentication method list that is tied to the WLAN. The **ip radius source-interface VLAN260** command is used for this server group and VLAN 260 does not specify an MTU meaning it is using the MTU of 1500. Just to confirm, the MTU of 1500 you can use the **show run all** command and look for the interface in the output.

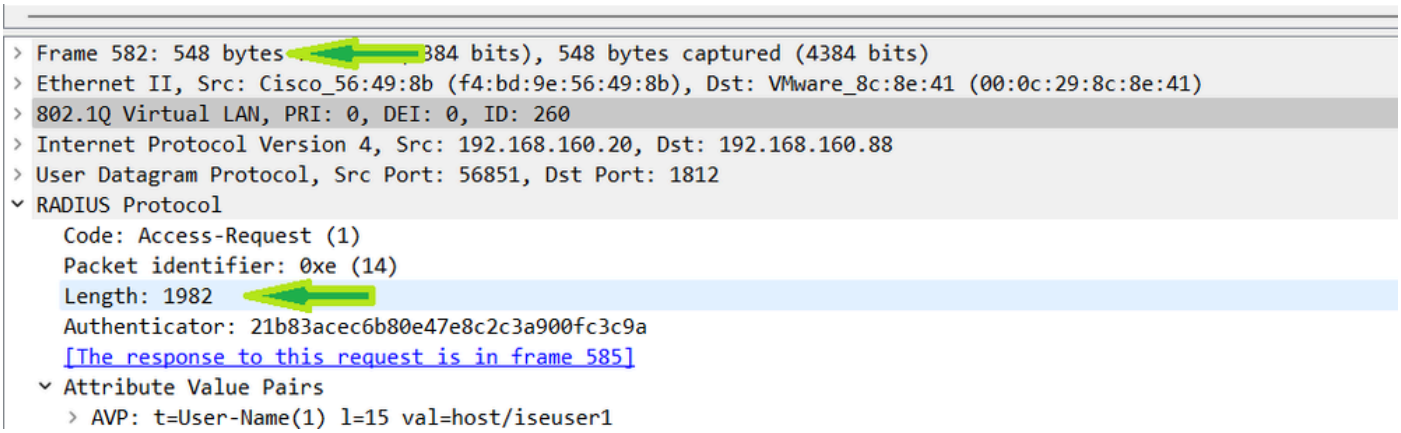
```
interface Vlan260
 ip address 192.168.160.20 255.255.255.0
 no ip clear-dont-fragment
 ip redirects
 ip unreachables
 no ip proxy-arp
 ip mtu 1500
```

Now look at the packet where the client cert has to be sent to ISE once the WLC adds the RADIUS data:



Note: Here, the bytes on the line is 1518. This is including headers outside of the Ethernet payload like the VLAN header and the layer 2 header. These are not counted towards the MTU.

```
> Frame 581: 1518 bytes c      (12144 bits), 1518 bytes captured (12144 bits)
> Ethernet II, Src: Cisco_56:49:8b (f4:bd:9e:56:49:8b), Dst: VMware_8c:8e:41 (00:0c:29:8c:8e:41)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 260
> Internet Protocol Version 4, Src: 192.168.160.20, Dst: 192.168.160.88
v Data (1480 bytes)
    Data: de13071407c63226010e07be21b83acec6b80e47e8c2c3a900fc3c9a010f686f73742f69…
    [Length: 1480]
```



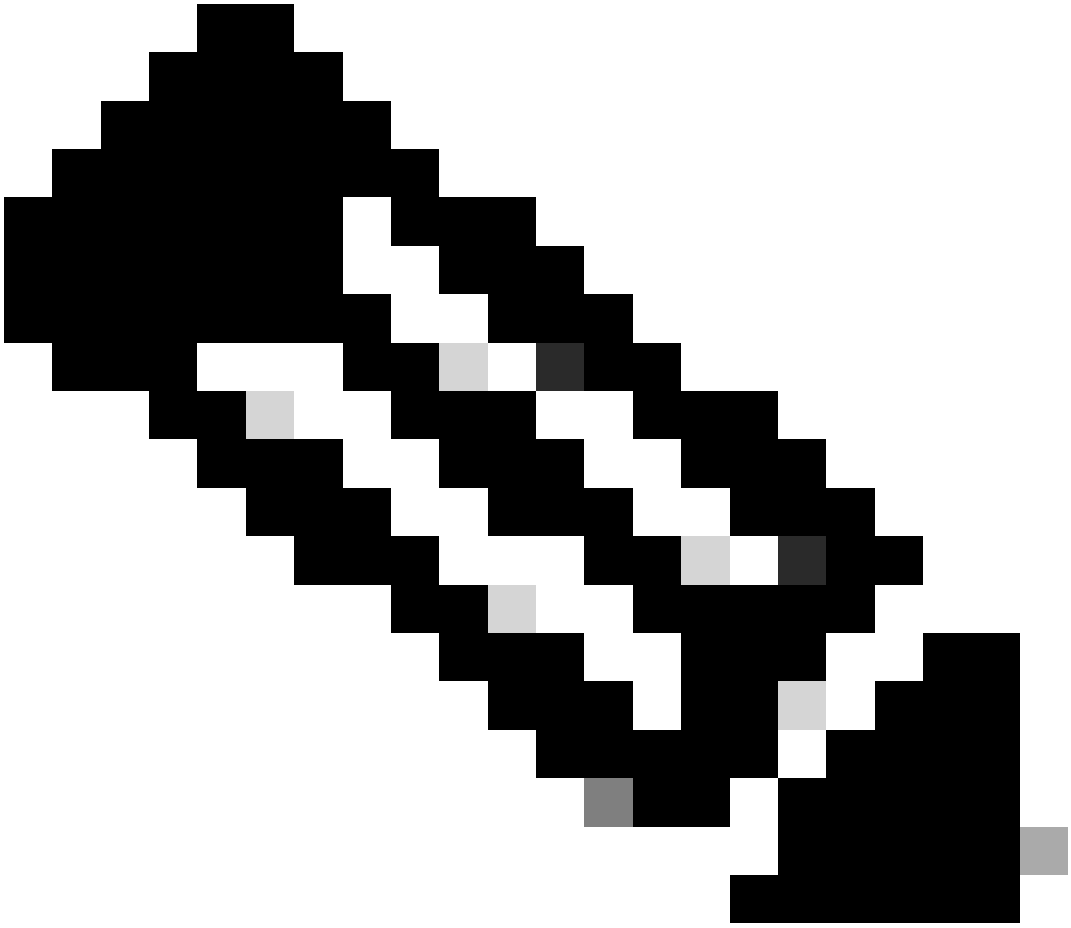
Here, you can see the data portion is fragmented at 1480. You can get that fragment under the 1500 MTU on the WMI. The next packet is under 550 bytes but you can see the total size of the RADIUS data is 1982. That said, fragmenting with the new MTU now works.

Using A Non-WMI Interface With An MTU Of 1200

Now, assume you want to fragment at a smaller MTU but do not want this change to affect any other traffic. No problem here, the config stays the same only the source-interface config is going to point to an SVI that was created just for this purpose. Change the method list to point to this new server group and this server group uses a source-interface that is not my WMI and has the MTU set to 1200. Here is what the config looks like:

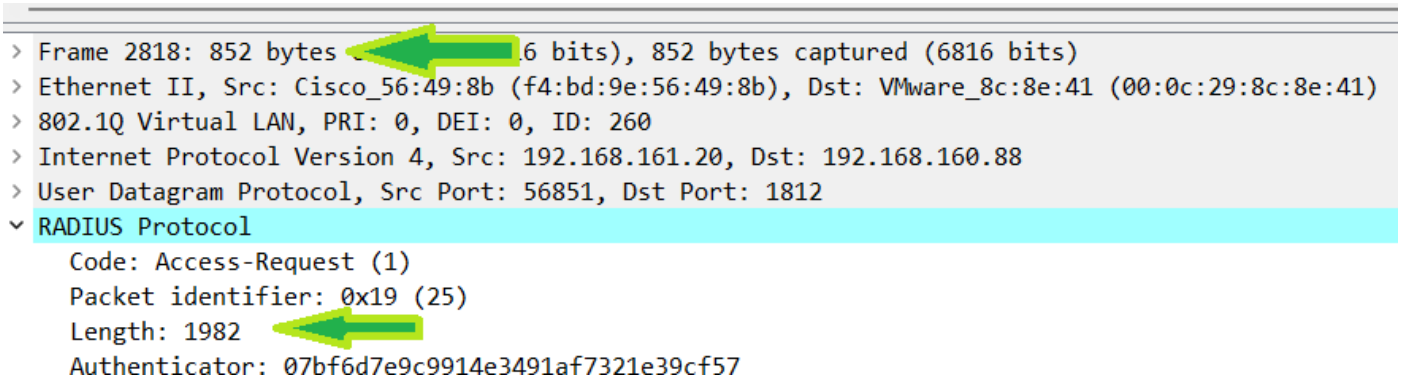
```
9800#show run aaa
!
aaa authentication dot1x TLS-AuthC group MTU1200
!
!
radius server ISE
 address ipv4 192.168.160.10 auth-port 1812 acct-port 1813
 key 6 _`gINMNXObFibbBMhNMFAbKUAAB
!
aaa group server radius MTU1200
 server name ISE
 ip radius source-interface Vlan261
 deadtime 5
!
9800#show run inter vlan 261
!
interface Vlan261
 ip address 192.168.161.20 255.255.255.0
 no ip proxy-arp
 ip mtu 1200
end
```

Next, see how the packets look with this lower MTU.



Note: Lowering the MTU and changing the fragmentation point is not part of the new behavior. This has always been true. If the default behavior of fragmenting at 1396 does not fit under the MTU, you would always fragment at a different point. It is part of this section just to help explain the options available.

```
> Frame 2817: 1214 bytes c                2 bits), 1214 bytes captured (9712 bits)
> Ethernet II, Src: Cisco_56:49:8b (f4:bd:9e:56:49:8b), Dst: VMware_8c:8e:41 (00:0c:29:8c:8e:41)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 260
> Internet Protocol Version 4, Src: 192.168.161.20, Dst: 192.168.160.88
v Data (1176 bytes)
    Data: de13071407c6b995011907be07bf6d7e9c9914e3491af7321e39cf57010f686f73742f69…
    [Length: 1176]
```



Here, the RADIUS data is still 1982 bytes but this time the data was fragmented at 1176 instead of the default 1376 it would have fragmented at if the source interface was not used. Remember that when you set the MTU to 1500 and use the source-interface command, you fragment at 1480. Using the config here allows you to manipulate the traffic to a lower MTU without interfering with other traffic on the WLC.

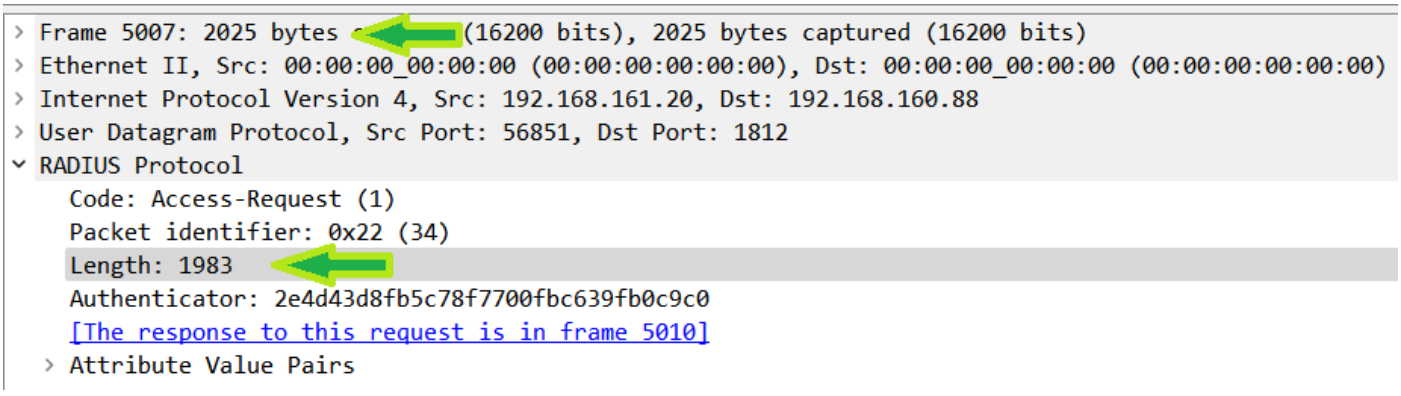
Using an MTU Of 9000 For Jumbo Frames

Since the feature was made for the option to send jumbo frames, it would be a shame to not test that as well still using the non-WMI interface of VLAN 261. However, now the IP MTU is set to 9000. One quick note, in order to be able to set the IP MTU on the SVI, you have to set the MTU to something higher than the IP MTU. You can see this in this config:

```
9800(config-if)#do sho run inter vl 261
!
interface Vlan261
 mtu 9100
 ip address 192.168.161.20 255.255.255.0
 no ip proxy-arp
 ip mtu 9000
end
```

Here, looking at the capture, you can see that the packet was never fragmented. It was sent as one whole packet with the RADIUS data size at 1983. Keep in mind, for this to work the rest of the network needs to be configured to allow a packet this size through.

Another thing to notice here is that the client MTU has not changed so the client is still fragmenting the EAP packet at 1492. The difference is that the WLC can add all the RADIUS data needed to send the packet to ISE without having to fragment the client data.



Conclusion

When you use EAP-TLS, the client is expected to send its certificate to the AAA server. These certificates are usually larger than the MTU so the client has to fragment it. The point at which the client fragments the data is pretty close to the MTU. Since the AP has to add the CAPWAP header, what the client is sending has to be fragmented. The WLC receives these two packets, puts them back together but then has to fragment it again to add the RADIUS data. At this point, the network admin is given some control over how the WLC fragments the EAP packet the client has sent.

If you add the **ip radius source-interface <interface you want to use>** command to the AAA server group, the WLC uses whatever interface you put there instead of (or including) the WMI. Using this command also tells the WLC to fragment at whatever the MTU of that interface is instead of the default 1396. This way, you have more control over how packets are moving through the network.

When using Cisco Catalyst Center, the source interface command is added to the server group, thus changing the default behavior.