EAP Fragmentation Implementations and Behavior

Contents

Introduction

This document describes how to understand and troubleshoot Extensible Authentication Protocol (EAP) sessions.

Background Information

Sections of this document are dedicated to coverage in these areas:

- Behavior of Authentication, Authorization, and Accounting (AAA) servers when they return the Server Certificate for the Extensible Authentication Protocol-Transport Layer Security (EAP-TLS) session
- Behavior of supplicants when they return the Client Certificate for the EAP-TLS session
- Interoperability when both the Microsoft Windows Native Supplicant and the Cisco AnyConnect Network Access Manager (NAM) are used
- Fragmentation in IP, RADIUS, and EAP-TLS and re-assembly process performed by network access devices
- The RADIUS Framed-Maximum Transmission Unit (MTU) attribute
- AAA servers' behavior when they perform fragmentation of EAP-TLS packets

Prerequisites

Requirements

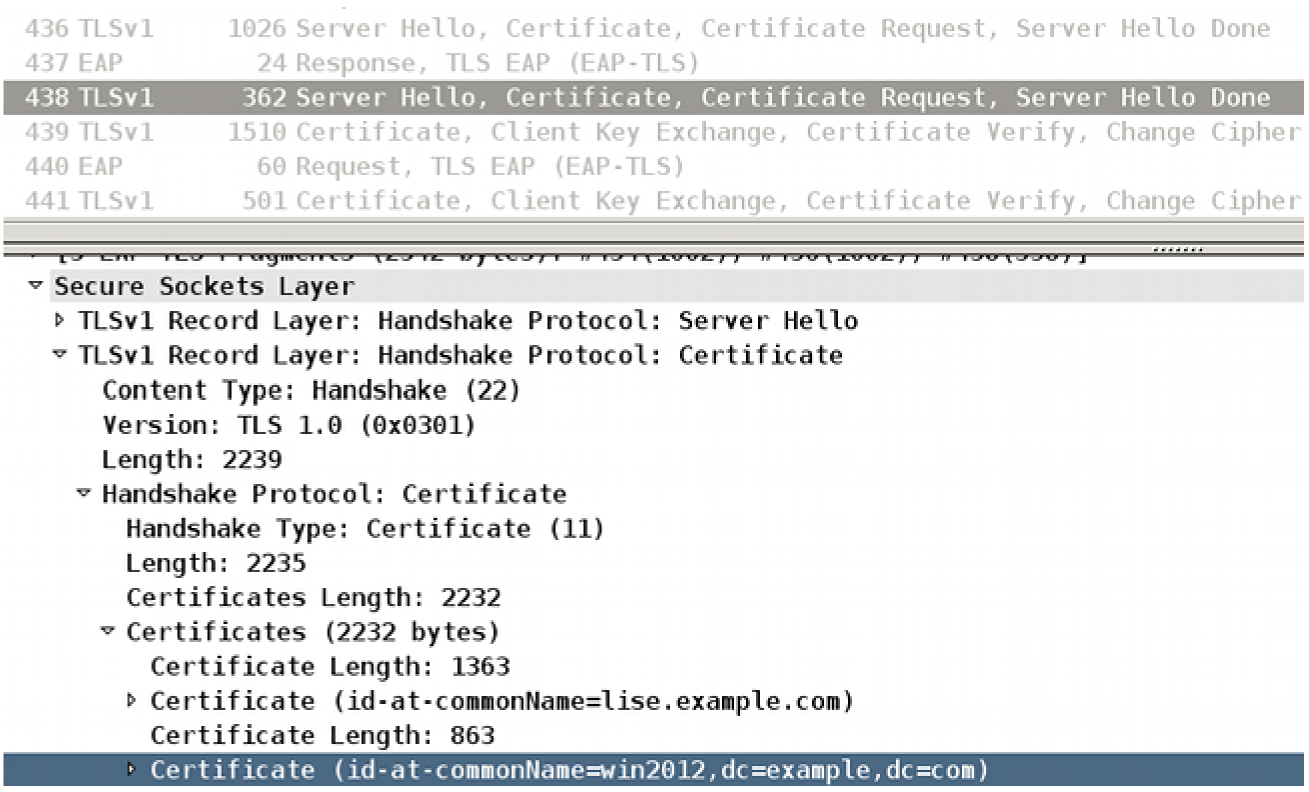
Cisco recommends that you have knowledge of these topics:

- EAP and EAP-TLS protocols
- Configuration of the Cisco Identity Services Engine (ISE)
- CLI configuration of Cisco Catalyst switches

It is necessary to have a good understanding of EAP and EAP-TLS in order to understand this article.

Certificate Chain Returned by the Server

The AAA server (Access Control Server (ACS) and ISE) always returns the full chain for the EAP-TLS packet with the Server Hello and the Server Certificate:



```
436 TLSv1    1026 Server Hello, Certificate, Certificate Request, Server Hello Done
437 EAP        24 Response, TLS EAP (EAP-TLS)
438 TLSv1     362 Server Hello, Certificate, Certificate Request, Server Hello Done
439 TLSv1    1510 Certificate, Client Key Exchange, Certificate Verify, Change Cipher
440 EAP        60 Request, TLS EAP (EAP-TLS)
441 TLSv1     501 Certificate, Client Key Exchange, Certificate Verify, Change Cipher

▿ Secure Sockets Layer
  ▹ TLSv1 Record Layer: Handshake Protocol: Server Hello
  ▿ TLSv1 Record Layer: Handshake Protocol: Certificate
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 2239
    ▿ Handshake Protocol: Certificate
        Handshake Type: Certificate (11)
        Length: 2235
        Certificates Length: 2232
      ▿ Certificates (2232 bytes)
          Certificate Length: 1363
        ▹ Certificate (id-at-commonName=lise.example.com)
          Certificate Length: 863
        ▹ Certificate (id-at-commonName=win2012,dc=example,dc=com)
```

The ISE identity certificate (Common Name (CN)=lise.example.com) is returned along with Certificate Authority (CA) that signed the CN=win2012,dc=example,dc=com. The behavior is the same for both ACS and ISE.

Certificate Chain Returned by the Supplicant

Microsoft Windows Native Supplicant

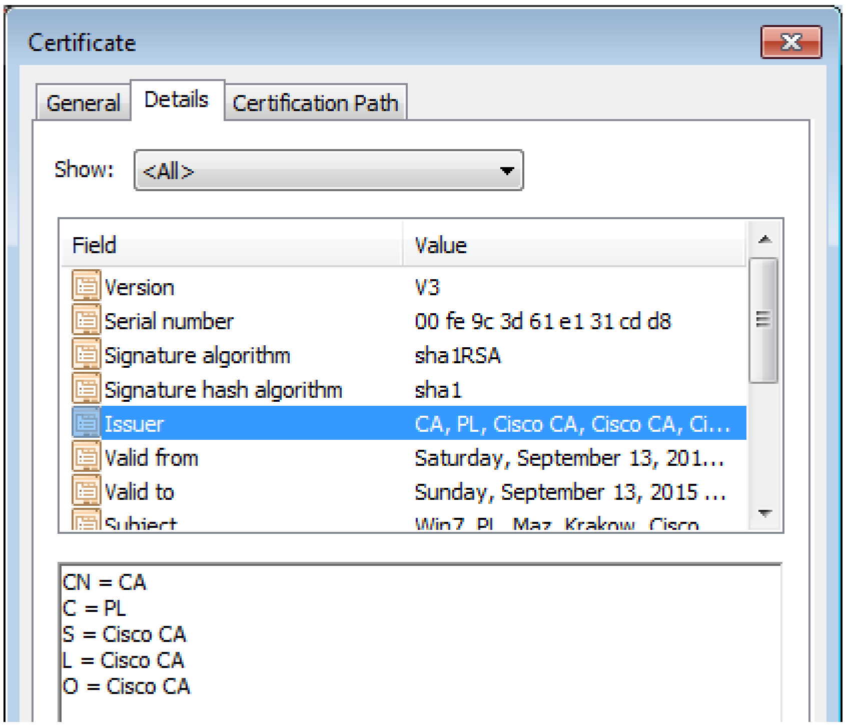
Microsoft Windows 7 Native supplicant configured in order to use EAP-TLS, with or without the "Simple certificate selection", does not send the full chain of the client certificate.

This behavior occurs even when client certificate is signed by a different CA (different chain) than the server certificate.

This example is related to the Server Hello and Certificate presented in the previous screenshot.

For that scenario, the ISE certificate is signed by the CA with the use of a subject name, CN=win2012,dc=example,dc=com.

But the user certificate installed in the Microsoft store is signed by a different CA, CN=CA,C=PL,S=Cisco CA,L=Cisco CA, O=Cisco CA.

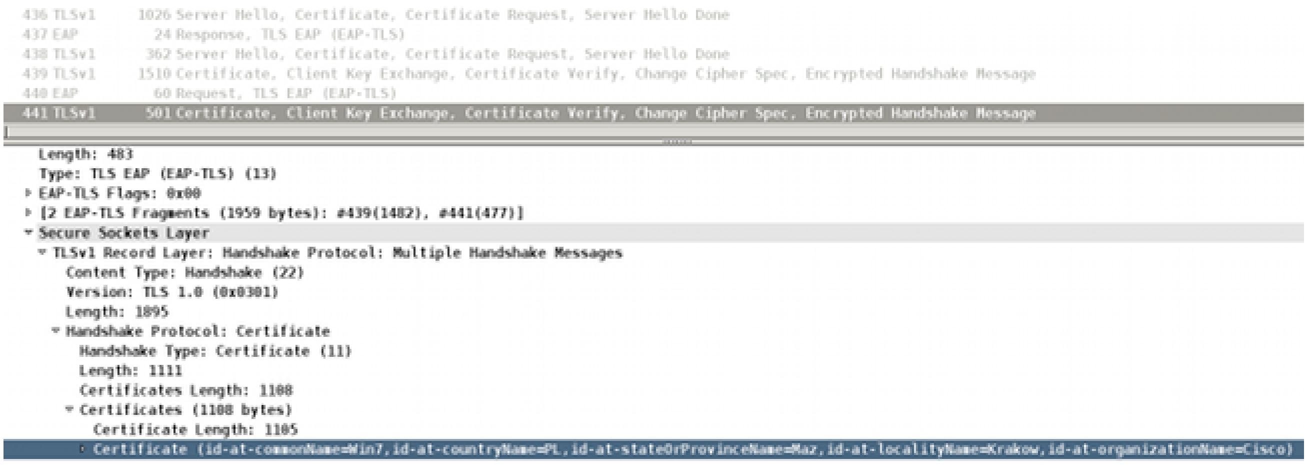


Show: <All>

Field	Value
Version	V3
Serial number	00 fe 9c 3d 61 e1 31 cd d8
Signature algorithm	sha1RSA
Signature hash algorithm	sha1
Issuer	CA, PL, Cisco CA, Cisco CA, Ci...
Valid from	Saturday, September 13, 201...
Valid to	Sunday, September 13, 2015 ...
Subject	Win7, PL, Maz, Krakow, Cisco

```
CN = CA
C = PL
S = Cisco CA
L = Cisco CA
O = Cisco CA
```

As a result, the Microsoft Windows supplicant responds with the client certificate only. The CA that signs it (CN=CA,S=PL,S=Cisco CA, L=Cisco CA, O=Cisco CA) is not attached.



```
436 TLSv1     1026 Server Hello, Certificate, Certificate Request, Server Hello Done
437 EAP         24 Response, TLS EAP (EAP-TLS)
438 TLSv1      362 Server Hello, Certificate, Certificate Request, Server Hello Done
439 TLSv1     1510 Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
440 EAP         60 Request, TLS EAP (EAP-TLS)
441 TLSv1      501 Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message

    Length: 483
    Type: TLS EAP (EAP-TLS) (13)
  ▸ EAP-TLS Flags: 0x00
  ▸ [2 EAP-TLS Fragments (1959 bytes): #439(1482), #441(477)]
  ▾ Secure Sockets Layer
    ▾ TLSv1 Record Layer: Handshake Protocol: Multiple Handshake Messages
        Content Type: Handshake (22)
        Version: TLS 1.0 (0x0301)
        Length: 1895
      ▾ Handshake Protocol: Certificate
          Handshake Type: Certificate (11)
          Length: 1111
          Certificates Length: 1108
        ▾ Certificates (1108 bytes)
            Certificate Length: 1105
          ▸ Certificate (id-at-commonName=Win7,id-at-countryName=PL,id-at-stateOrProvinceName=Maz,id-at-localityName=Krakow,id-at-organizationName=Cisco)
```

Because of this behavior, the AAA servers possibly encounter problems when they validate client certificates. The example relates to Microsoft Windows 7 SP1 Professional.

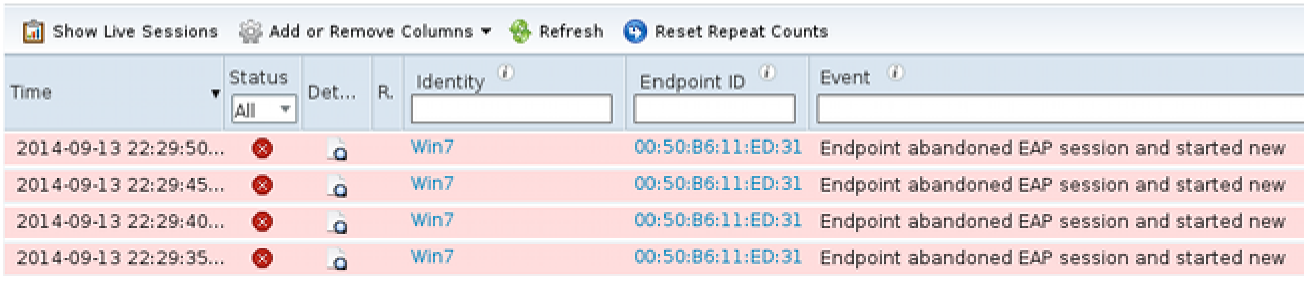
Solution

A full certificate chain is to be installed on the certificate store of ACS and ISE (all CA and sub CA signing client certificates).

Problems with certificate validation can be easily detected on ACS or ISE. The information about untrusted certificate is presented and ISE reports:

```
12514 EAP-TLS failed SSL/TLS handshake because of an unknown CA in the client
certificates chain
```

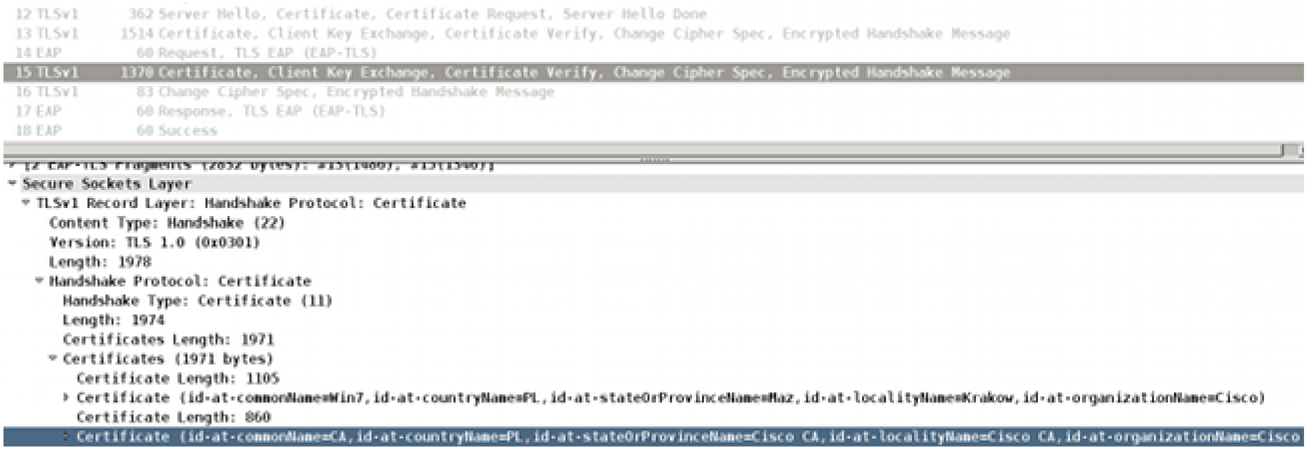
Problems with certificate validation on the supplicant are not easily detectable. Usually the AAA server responds that "Endpoint abondoned EAP session":



Time	Status All	Det...	R.	Identity	Endpoint ID	Event
2014-09-13 22:29:50...	❌	🔍		Win7	00:50:B6:11:ED:31	Endpoint abandoned EAP session and started new
2014-09-13 22:29:45...	❌	🔍		Win7	00:50:B6:11:ED:31	Endpoint abandoned EAP session and started new
2014-09-13 22:29:40...	❌	🔍		Win7	00:50:B6:11:ED:31	Endpoint abandoned EAP session and started new
2014-09-13 22:29:35...	❌	🔍		Win7	00:50:B6:11:ED:31	Endpoint abandoned EAP session and started new

AnyConnect NAM

The AnyConnect NAM does not have this limitation. In the same scenario, it attaches the complete chain of the client certificate (the correct CA is attached):



```
12 TLSv1      362 Server Hello, Certificate, Certificate Request, Server Hello Done
13 TLSv1     1514 Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
14 EAP         60 Request, TLS EAP (EAP-TLS)
15 TLSv1     1370 Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
16 TLSv1       83 Change Cipher Spec, Encrypted Handshake Message
17 EAP         60 Response, TLS EAP (EAP-TLS)
18 EAP         60 Success

▸ [2 EAP-TLS Fragments (2852 bytes): #13(1460), #15(1340)]
▾ Secure Sockets Layer
  ▾ TLSv1 Record Layer: Handshake Protocol: Certificate
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 1978
    ▾ Handshake Protocol: Certificate
        Handshake Type: Certificate (11)
        Length: 1974
        Certificates Length: 1971
      ▾ Certificates (1971 bytes)
          Certificate Length: 1105
        ▸ Certificate (id-at-commonName=Win7,id-at-countryName=PL,id-at-stateOrProvinceName=Maz,id-at-localityName=Krakow,id-at-organizationName=Cisco)
          Certificate Length: 860
        ▸ Certificate (id-at-commonName=CA,id-at-countryName=PL,id-at-stateOrProvinceName=Cisco CA,id-at-localityName=Cisco CA,id-at-organizationName=Cisco
```

Microsoft Windows Native Supplicant Along with AnyConnect NAM

When both services are up, AnyConnect NAM takes precedence.

Even when the NAM service does not run, it still hooks on the Microsoft Windows API and forwards the EAP packets, which can cause problems for the Microsoft Windows Native supplicant.

Here is an example of such a failure.

You enable tracing on Microsoft Windows with this command:

```
C:\netsh ras set tracing * enable
```

The traces (**c:\windows\trace\svchost_RASTLS.LOG**) show:

<#root>

```
[2916] 09-14 21:29:11:254: >> Received Request (Code: 1) packet: Id: 55, Length:
6, Type: 13, TLS blob length: 0. Flags: S
[2916] 09-14 21:29:11:254: << Sending Response (Code: 2) packet: Id: 55, Length:
105, Type: 13, TLS blob length: 95. Flags: L
[1804] 09-14 21:29:11:301: >> Received Request (Code: 1) packet: Id: 56, Length:
1012, Type: 13, TLS blob length: 2342. Flags: LM
[1804] 09-14 21:29:11:301: << Sending Response (Code: 2) packet: Id: 56, Length:
6, Type: 13, TLS blob length: 0. Flags:
[1804] 09-14 21:29:11:348: >> Received Request (Code: 1) packet: Id: 57, Length:
1008, Type: 13, TLS blob length: 0. Flags: M
[1804] 09-14 21:29:11:348: << Sending Response (Code: 2) packet: Id: 57, Length:
6, Type: 13, TLS blob length: 0. Flags:
[1804] 09-14 21:29:11:363: >> Received Request (Code: 1) packet: Id: 58, Length:
344, Type: 13, TLS blob length: 0. Flags:
[1804] 09-14 21:29:11:363: << Sending Response (Code: 2) packet: Id: 58, Length:
1492, Type: 13, TLS blob length: 1819. Flags: LM
[3084] 09-14 21:31:11:203: >> Received Request (Code: 1) packet: Id: 122, Length:
6, Type: 13, TLS blob length: 0. Flags: S
[3084] 09-14 21:31:11:218: << Sending Response (Code: 2) packet: Id: 122, Length:
105, Type: 13, TLS blob length: 95. Flags: L
[3420] 09-14 21:31:11:249: >> Received Request (Code: 1) packet: Id: 123, Length:
1012, Type: 13, TLS blob length: 2342. Flags: LM
[3420] 09-14 21:31:11:249: << Sending Response (Code: 2) packet: Id: 123, Length:
6, Type: 13, TLS blob length: 0. Flags:
[3420] 09-14 21:31:11:281: >> Received Request (Code: 1) packet: Id: 124, Length:
1008, Type: 13, TLS blob length: 0. Flags: M
[3420] 09-14 21:31:11:281: << Sending Response (Code: 2) packet: Id: 124, Length:
6, Type: 13, TLS blob length: 0. Flags:
[3420] 09-14 21:31:11:281: >> Received Request (Code: 1) packet: Id: 125, Length:
344, Type: 13, TLS blob length: 0. Flags:
[3420] 09-14 21:31:11:296: <<
```

`Sending Response (Code: 2)`

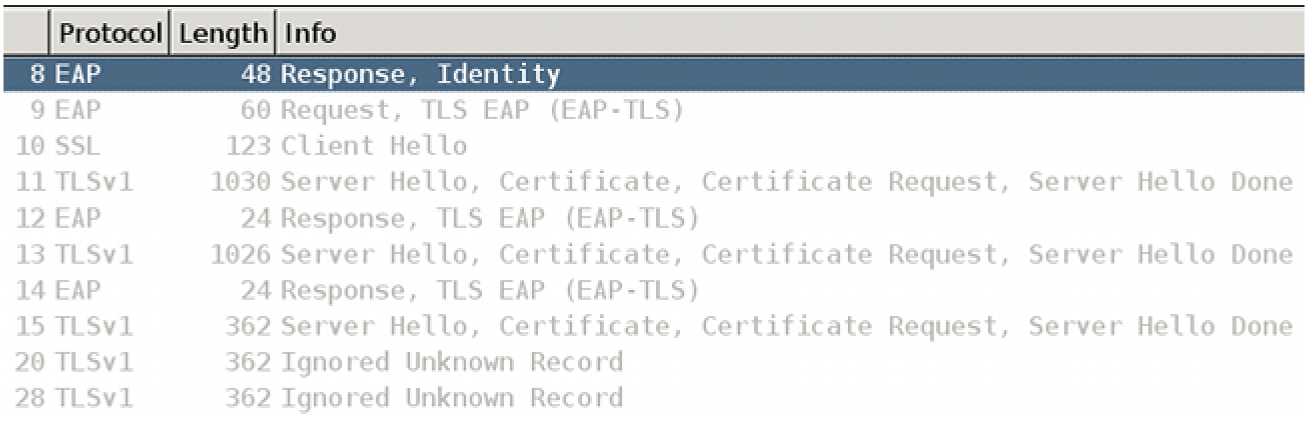
` packet: Id: 125, Length:`

`1492`

`, Type: 13,`

`TLS blob length: 1819. Flags: LM`

The last packet is a Client Certificate (EAP-TLS fragment 1 with EAP size 1492) sent by the Microsoft Windows Native supplicant. Unfortunately, Wireshark does not show that packet:



```
   Protocol Length Info
 8 EAP          48 Response, Identity
 9 EAP          60 Request, TLS EAP (EAP-TLS)
10 SSL         123 Client Hello
11 TLSv1      1030 Server Hello, Certificate, Certificate Request, Server Hello Done
12 EAP          24 Response, TLS EAP (EAP-TLS)
13 TLSv1      1026 Server Hello, Certificate, Certificate Request, Server Hello Done
14 EAP          24 Response, TLS EAP (EAP-TLS)
15 TLSv1       362 Server Hello, Certificate, Certificate Request, Server Hello Done
20 TLSv1       362 Ignored Unknown Record
28 TLSv1       362 Ignored Unknown Record
```

And that packet is not really sent; the last one was the third fragment of the EAP-TLS carrying server certificate.

It has been consumed by the AnyConnect NAM module that hooks on the Microsoft Windows API.

That is why it is not advised to use AnyConnect along with the Microsoft Windows Native supplicant.

When you use any AnyConnect services, it is advised to use NAM also (when 802.1x services are needed), not the Microsoft Windows Native Supplicant.

Fragmentation

The fragmentation possibly occurs on multiple layers:

- IP
- RADIUS Attribute Value Pairs (AVP)
- EAP-TLS

Cisco IOS® switches are very intelligent. They can understand EAP and EAP-TLS formats.

Although the switch is not able to decrypt the TLS tunnel, it is responsible for fragmentation, and assembly and re-assembly of the EAP packets when encapsulation in Extensible Authentication Protocol over LAN (EAPoL) or RADIUS.

EAP protocol does not support fragmentation. Here is an excerpt from RFC 3748 (EAP):

"Fragmentation is not supported within EAP itself; however, individual EAP methods may support this."

EAP-TLS is such an example. Here is an excerpt from RFC 5216 (EAP-TLS), section 2.1.5 (fragmentation):

"When an EAP-TLS peer receives an EAP-Request packet with the M bit set, it MUST respond with an EAP-Response with EAP-Type=EAP-TLS and no data.

This serves as a fragment ACK. **The EAP server MUST wait until it receives the EAP-Response before sending another fragment**."

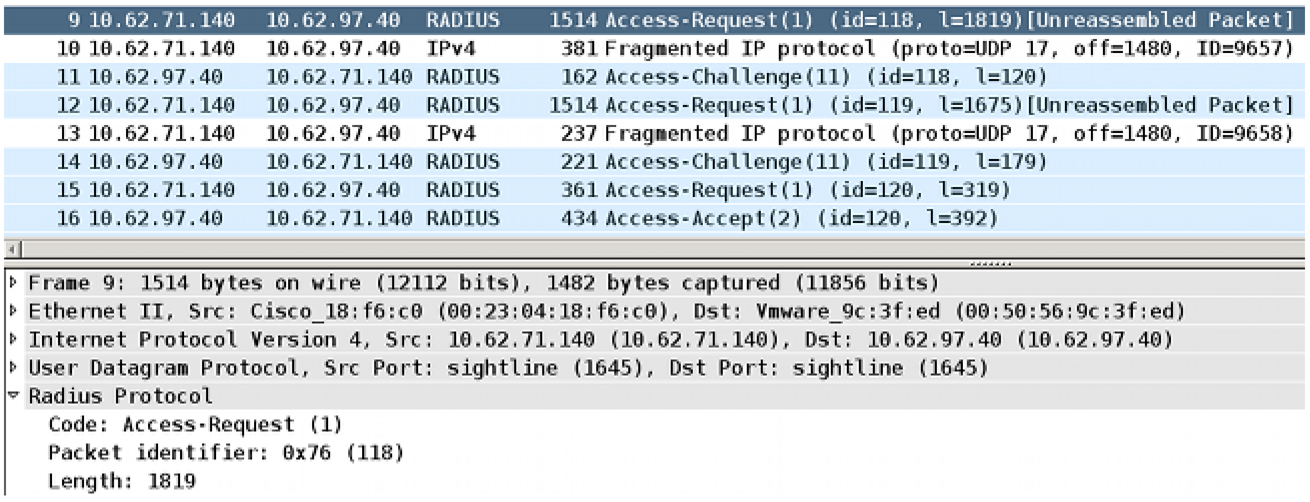
The last sentence describes a very important feature of AAA servers. They must wait for the ACK before they can send another EAP fragment. A similar rule is used for the supplicant:

"The EAP peer MUST wait until it receives the EAP-Request before sending another fragment."

Fragmentation in the IP Layer

Fragmentation can occur only between the Network Access Device (NAD) and the AAA server (IP/UDP/RADIUS used as a transport).

This situation occurs when NAD (Cisco IOS switch) tries to send the RADIUS Request that contains the EAP payload, which is bigger then MTU of the interface:



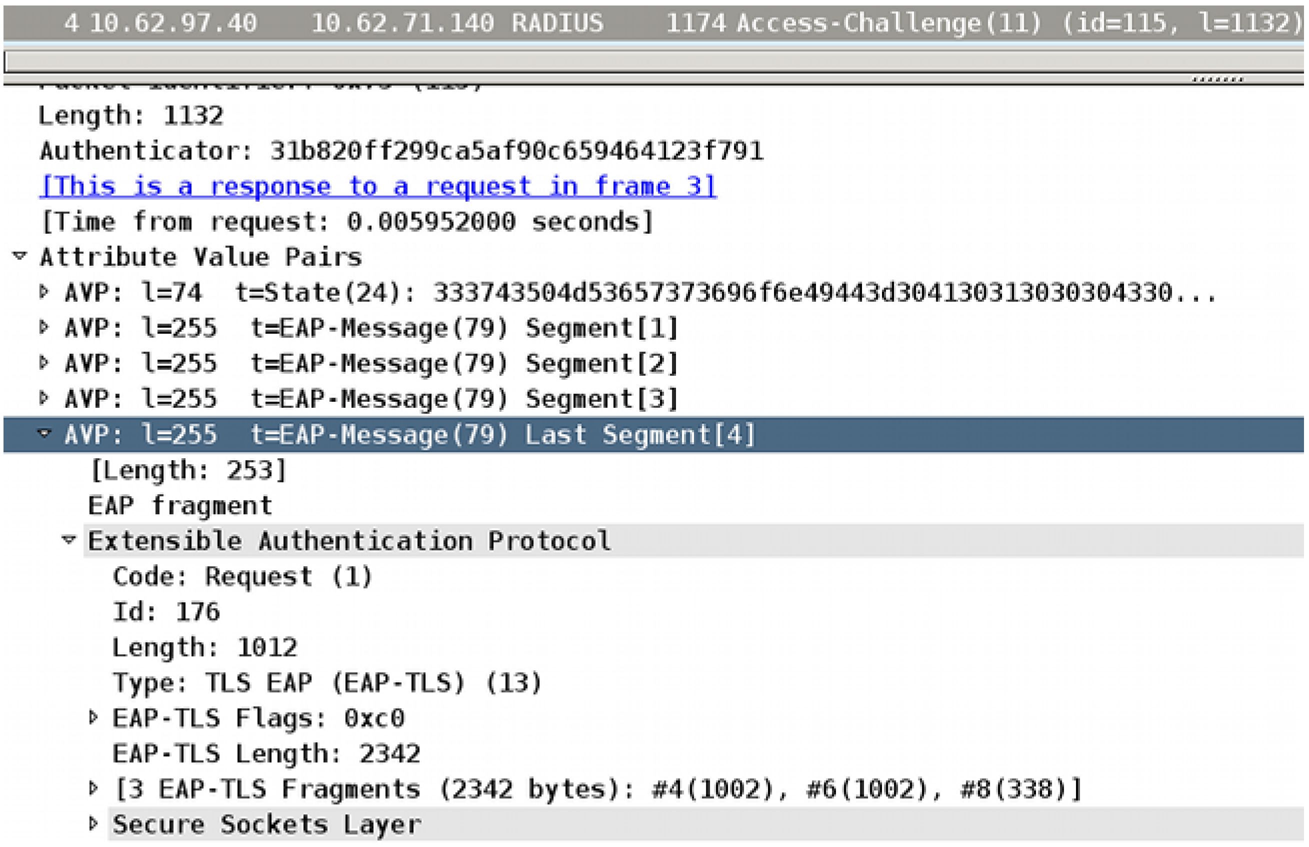
```
 9 10.62.71.140   10.62.97.40    RADIUS   1514 Access-Request(1) (id=118, l=1819)[Unreassembled Packet]
10 10.62.71.140   10.62.97.40    IPv4      381 Fragmented IP protocol (proto=UDP 17, off=1480, ID=9657)
11 10.62.97.40    10.62.71.140   RADIUS    162 Access-Challenge(11) (id=118, l=120)
12 10.62.71.140   10.62.97.40    RADIUS   1514 Access-Request(1) (id=119, l=1675)[Unreassembled Packet]
13 10.62.71.140   10.62.97.40    IPv4      237 Fragmented IP protocol (proto=UDP 17, off=1480, ID=9658)
14 10.62.97.40    10.62.71.140   RADIUS    221 Access-Challenge(11) (id=119, l=179)
15 10.62.71.140   10.62.97.40    RADIUS    361 Access-Request(1) (id=120, l=319)
16 10.62.97.40    10.62.71.140   RADIUS    434 Access-Accept(2) (id=120, l=392)

▸ Frame 9: 1514 bytes on wire (12112 bits), 1482 bytes captured (11856 bits)
▸ Ethernet II, Src: Cisco_18:f6:c0 (00:23:04:18:f6:c0), Dst: Vmware_9c:3f:ed (00:50:56:9c:3f:ed)
▸ Internet Protocol Version 4, Src: 10.62.71.140 (10.62.71.140), Dst: 10.62.97.40 (10.62.97.40)
▸ User Datagram Protocol, Src Port: sightline (1645), Dst Port: sightline (1645)
▾ Radius Protocol
    Code: Access-Request (1)
    Packet identifier: 0x76 (118)
    Length: 1819
```

Most Cisco IOS versions are not intelligent enough and do not try to assemble EAP packets received via EAPoL and combine them in a RADIUS packet that can fit in the MTU of the physical interface towards the AAA server.

AAA servers are more intelligent (as presented in the next sections).

Fragmentation in RADIUS

This is not really any kind of fragmentation. As per RFC 2865, a single RADIUS attribute can have up to 253 bytes of data.Because of that, the EAP payload is always transmitted in multiple EAP-Message RADIUS attributes:



```
  4 10.62.97.40    10.62.71.140 RADIUS    1174 Access-Challenge(11) (id=115, l=1132)
  Length: 1132
  Authenticator: 31b820ff299ca5af90c659464123f791
  [This is a response to a request in frame 3]
  [Time from request: 0.005952000 seconds]
▽ Attribute Value Pairs
  ▷ AVP: l=74  t=State(24): 333743504d53657373696f6e49443d3041303130303304330...
  ▷ AVP: l=255  t=EAP-Message(79) Segment[1]
  ▷ AVP: l=255  t=EAP-Message(79) Segment[2]
  ▷ AVP: l=255  t=EAP-Message(79) Segment[3]
  ▽ AVP: l=255  t=EAP-Message(79) Last Segment[4]
      [Length: 253]
      EAP fragment
    ▽ Extensible Authentication Protocol
        Code: Request (1)
        Id: 176
        Length: 1012
        Type: TLS EAP (EAP-TLS) (13)
      ▷ EAP-TLS Flags: 0xc0
        EAP-TLS Length: 2342
      ▷ [3 EAP-TLS Fragments (2342 bytes): #4(1002), #6(1002), #8(338)]
      ▷ Secure Sockets Layer
```

Those EAP-Message attributes are re-assembled and interpreted by Wireshark (the "Last Segment" attribute reveals the payload of the whole EAP packet).

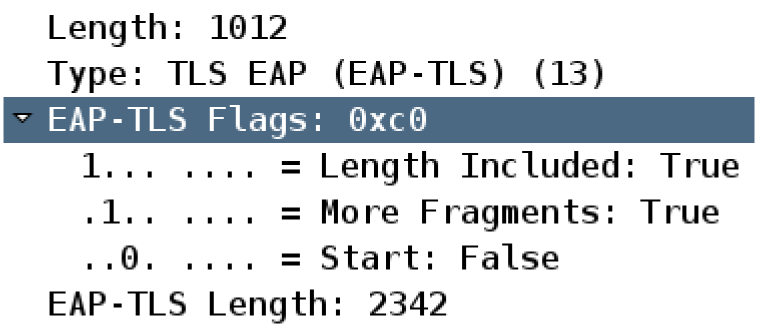
The Length header in the EAP packet is equal to1,012, and four RADIUS AVPs are required to transport it.

Fragmentation in EAP-TLS

From the same screenshot, you can see that:

- EAP packet length is 1,012
- EAP-TLS length is 2,342

This suggests that it is the first EAP-TLS fragment and the supplicant expects more, which can be confirmed if you examine the EAP-TLS flags:



```
  Length: 1012
  Type: TLS EAP (EAP-TLS) (13)
▽ EAP-TLS Flags: 0xc0
    1... .... = Length Included: True
    .1.. .... = More Fragments: True
    ..0. .... = Start: False
  EAP-TLS Length: 2342
```

This kind of fragmentation most frequently occurs in:

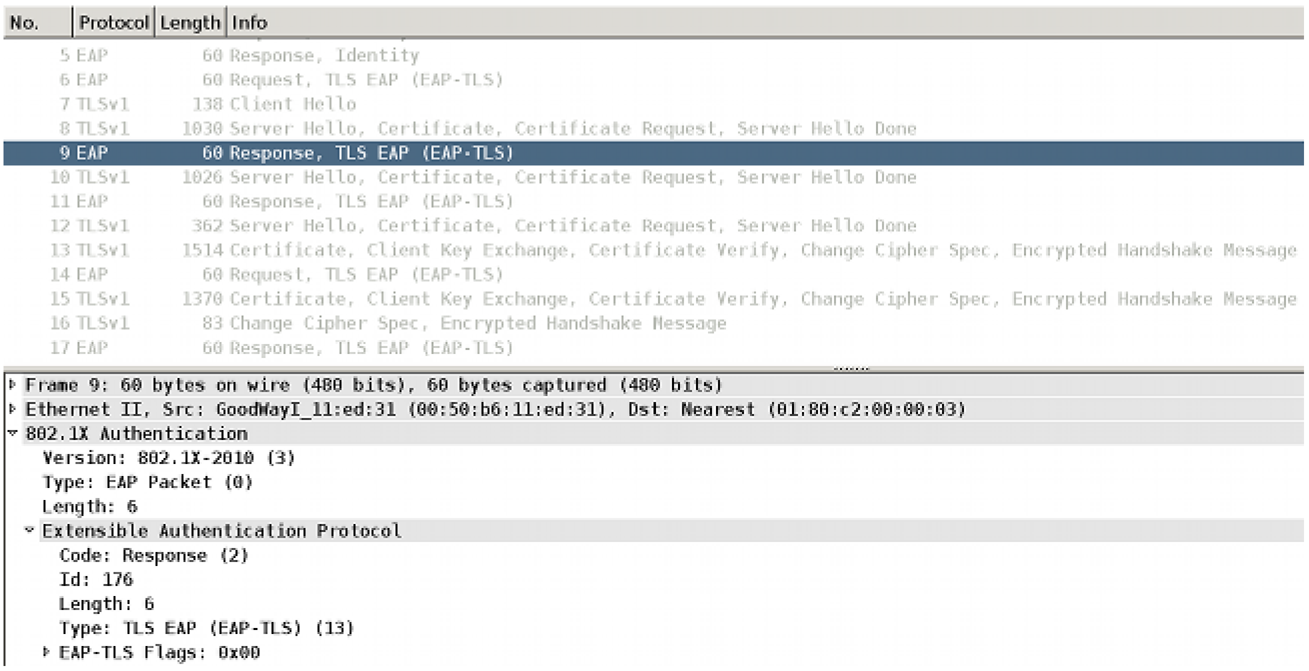
- RADIUS Access-Challenge sent by the AAA server, which carries the EAP-Request with the Secure Sockets Layer (SSL) Server Certificate with the whole chain.
- RADIUS Access-Request send by NAD, which carries the EAP-Response with the SSL Client

Certificate with the whole chain.

EAP-TLS Fragment Confirmation

As explained earlier, every EAP-TLS fragment must be acknowledged before subsequent fragments are sent.

Here is an example (packet captures for EAPoL between the supplicant and the NAD):

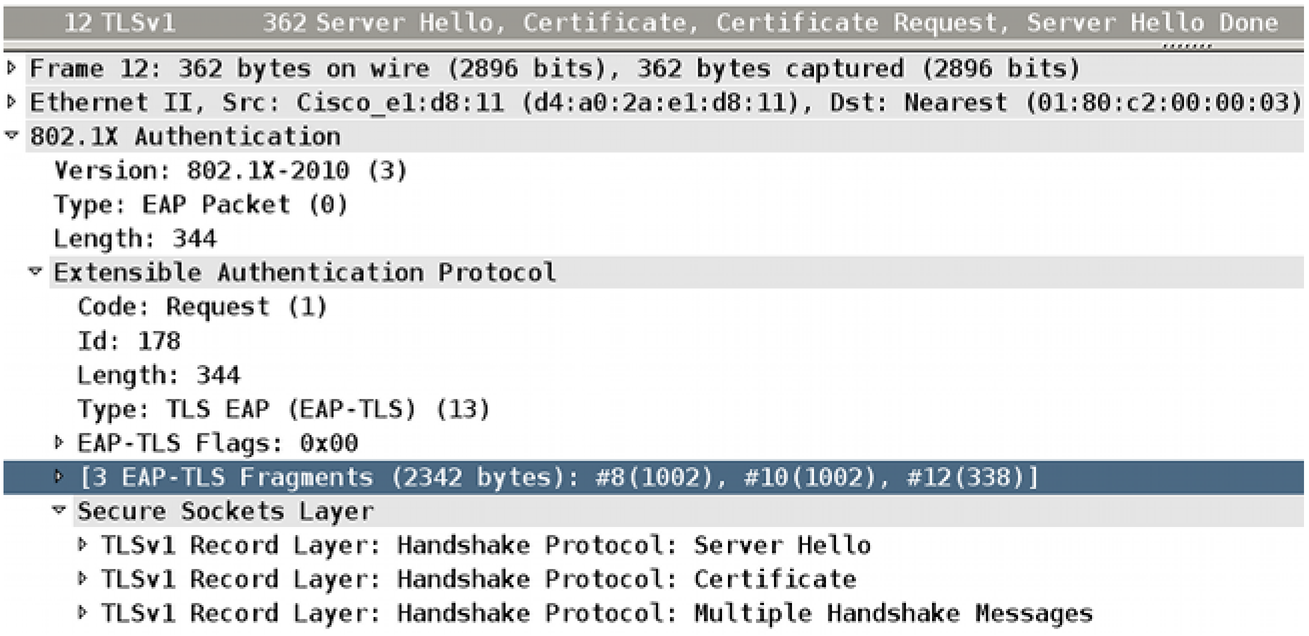


```
No.  Protocol Length Info
  5 EAP        60 Response, Identity
  6 EAP        60 Request, TLS EAP (EAP-TLS)
  7 TLSv1     138 Client Hello
  8 TLSv1    1030 Server Hello, Certificate, Certificate Request, Server Hello Done
  9 EAP        60 Response, TLS EAP (EAP-TLS)
 10 TLSv1    1026 Server Hello, Certificate, Certificate Request, Server Hello Done
 11 EAP        60 Response, TLS EAP (EAP-TLS)
 12 TLSv1     362 Server Hello, Certificate, Certificate Request, Server Hello Done
 13 TLSv1    1514 Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
 14 EAP        60 Request, TLS EAP (EAP-TLS)
 15 TLSv1    1370 Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
 16 TLSv1      83 Change Cipher Spec, Encrypted Handshake Message
 17 EAP        60 Response, TLS EAP (EAP-TLS)

▸ Frame 9: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
▸ Ethernet II, Src: GoodWayI_11:ed:31 (00:50:b6:11:ed:31), Dst: Nearest (01:80:c2:00:00:03)
▾ 802.1X Authentication
    Version: 802.1X-2010 (3)
    Type: EAP Packet (0)
    Length: 6
  ▾ Extensible Authentication Protocol
      Code: Response (2)
      Id: 176
      Length: 6
      Type: TLS EAP (EAP-TLS) (13)
    ▸ EAP-TLS Flags: 0x00
```

EAPoL frames and the AAA server return the server certificate:

- That certificate is sent in an EAP-TLS fragment (packet 8).
- The supplicant acknowledges that fragment (packet 9).
- The second EAP-TLS fragment is forwarded by NAD (packet 10).
- The supplicant acknowledges that fragment (packet 11).
- The third EAP-TLS fragment is forwarded by NAD (packet 12).
- The supplicant does not need to acknowledge this; rather, it proceeds with the client certificate that starts at packet 13.

Here are the details of packet 12:

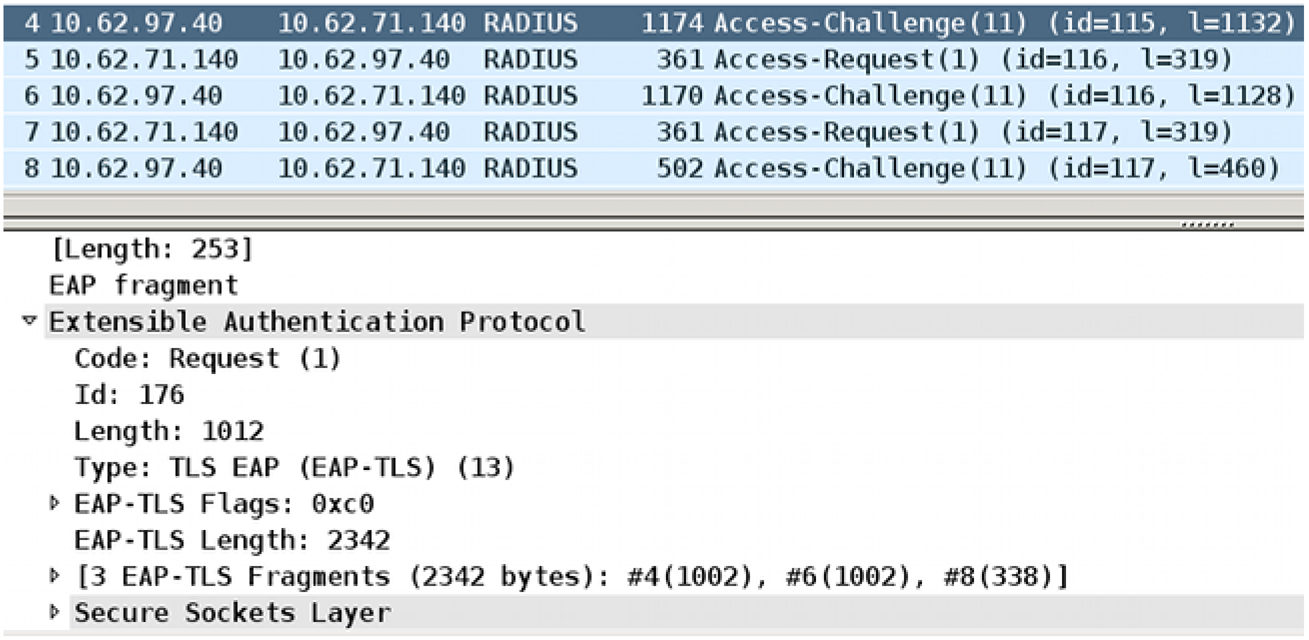


```
  12 TLSv1     362 Server Hello, Certificate, Certificate Request, Server Hello Done
▷ Frame 12: 362 bytes on wire (2896 bits), 362 bytes captured (2896 bits)
▷ Ethernet II, Src: Cisco_e1:d8:11 (d4:a0:2a:e1:d8:11), Dst: Nearest (01:80:c2:00:00:03)
▽ 802.1X Authentication
    Version: 802.1X-2010 (3)
    Type: EAP Packet (0)
    Length: 344
  ▽ Extensible Authentication Protocol
      Code: Request (1)
      Id: 178
      Length: 344
      Type: TLS EAP (EAP-TLS) (13)
    ▷ EAP-TLS Flags: 0x00
    ▷ [3 EAP-TLS Fragments (2342 bytes): #8(1002), #10(1002), #12(338)]
    ▽ Secure Sockets Layer
      ▷ TLSv1 Record Layer: Handshake Protocol: Server Hello
      ▷ TLSv1 Record Layer: Handshake Protocol: Certificate
      ▷ TLSv1 Record Layer: Handshake Protocol: Multiple Handshake Messages
```

You can see that Wireshark re-assembled packets 8, 10, and 12.

The size of the EAP fragments is1,002, 1,002, and 338, which brings the total size of the EAP-TLS message to 2342;

Total EAP-TLS message length is announced in every fragment. This can be confirmed if you examine RADIUS packets (between NAD and AAA server):



```
4 10.62.97.40   10.62.71.140 RADIUS   1174 Access-Challenge(11) (id=115, l=1132)
5 10.62.71.140  10.62.97.40  RADIUS    361 Access-Request(1) (id=116, l=319)
6 10.62.97.40   10.62.71.140 RADIUS   1170 Access-Challenge(11) (id=116, l=1128)
7 10.62.71.140  10.62.97.40  RADIUS    361 Access-Request(1) (id=117, l=319)
8 10.62.97.40   10.62.71.140 RADIUS    502 Access-Challenge(11) (id=117, l=460)

    [Length: 253]
    EAP fragment
  ▽ Extensible Authentication Protocol
      Code: Request (1)
      Id: 176
      Length: 1012
      Type: TLS EAP (EAP-TLS) (13)
    ▷ EAP-TLS Flags: 0xc0
      EAP-TLS Length: 2342
    ▷ [3 EAP-TLS Fragments (2342 bytes): #4(1002), #6(1002), #8(338)]
    ▷ Secure Sockets Layer
```

RADIUS packets 4, 6, and 8 carry those three EAP-TLS fragments. The first two fragments are acknowledged.

Wireshark is able to present the information about the EAP-TLS fragments (size: 1,002 + 1,002 + 338 = 2,342).

This scenario and example was easy. The Cisco IOS switch did not need to change the EAP-TLS fragment size.

EAP-TLS Fragments Re-assembled with Different Size

Consider what happens when NAD MTU towards AAA server is 9,000 bytes (jumbo frame) and the AAA server is also connected with the use of the interface that supports jumbo frames.

Most of the typical supplicants are connected with the use of a 1Gbit link with a MTU of 1,500.

In such a scenario, the Cisco IOS switch performs EAP-TLS "assymetric" assembly and re-assembly and changes EAP-TLS fragments sizes.

Here is an example for a large EAP message sent by the AAA server (SSL Server Certificate):

1. The AAA server must send an EAP-TLS message with a SSL Server Certificate. The total size of that EAP packet is 3,000. After it is encapsulated in RADIUS Access-Challenge/UDP/IP, it is still less than the AAA server interface MTU. A single IP packet is sent with 12 RADIUS EAP-Message attributes. There is no IP nor EAP-TLS fragmentation.

2. The Cisco IOS switch receives such a packet, decapsulates it, and decides that EAP needs to be sent via EAPoL to the supplicant. Since EAPoL does not support fragmentation, the switch must perform EAP-TLS fragmentation.

3. The Cisco IOS switch prepares the first EAP-TLS fragment that can fit into the MTU of the interface towards the supplicant (1,500).

4. This fragment is confirmed by the supplicant.

5. Another EAP-TLS fragment is sent after acknowledgement is received.

6. This fragment is confirmed by the supplicant.

7. The last EAP-TLS fragment is sent by the switch.

This scenario reveals that:

- Under some circumstances, the NAD must create EAP-TLS fragments.
- The NAD is responsible for sending/acknowledging those fragments.

The same situation can occur for a supplicant connected via a link that supports jumbo frames while the AAA server has a smaller MTU (then the Cisco IOS switch creates EAP-TLS fragments when it sends the EAP packet towards the AAA server).

RADIUS Attribute Framed-MTU

For RADIUS, there is a Framed-MTU attribute defined in RFC 2865:

"This Attribute indicates the Maximum Transmission Unit to be configured for the user, when it is not negotiated by some other means (such as PPP). It MAY be used in Access-Accept packets.

It MAY be used in an Access-Request packet as a hint by the NAS to the server that it would prefer that value, but the server is not required to honor the hint."

ISE does not honor the hint. The value of the Framed-MTU sent by NAD in the Access-Request does not have any impact on the fragmentation performed by ISE.

Multiple modern Cisco IOS switches do not allow changes to the MTU of the Ethernet interface except for jumbo frames settings enabled globally on the switch. The configuration of jumbo frames impacts the value of the Framed-MTU attribute sent in the RADIUS Access-Request. For example, you set:

```
<#root>
Switch(config)#
system mtu jumbo 9000
```

This forces the switch to send Framed-MTU = 9000 in all RADIUS Access-Requests. The same for the system MTU without jumbo frames:

```
<#root>
Switch(config)#
system mtu 1600
```

This forces the switch to send Framed-MTU = 1600 in all RADIUS Access-Requests.

Notice that modern Cisco IOS switches do not allow you to decrease the system MTU value below 1,500.

AAA Servers and Supplicant Behavior When You Send EAP Fragments

ISE

ISE always tries to send EAP-TLS fragments (usually Server Hello with Certificate) that are 1,002 bytes long (although the last fragment is usually smaller).

It does not honor the RADIUS Framed-MTU. It is not possible to reconfigure it to send bigger EAP-TLS fragments.

Microsoft Network Policy Server (NPS)

It is possible to configure the size of the EAP-TLS fragments if you configure the Framed-MTU attribute locally on NPS.

Event though the Configure the EAP Payload Size on Microsoft NPS article mentions that the default value of a framed MTU for the NPS RADIUS server is 1,500, the Cisco Technical Assistance Center (TAC) lab has shown that it sends 2,000 with the default settings (confirmed on a Microsoft Windows 2012 Datacenter).

It is tested that setting **Framed-MTU locally** as per the previously mentioned guide is respected by NPS, and it fragments the EAP messages into fragments of a size set in Framed-MTU. But the Framed-MTU attribute received in the Access-Request is not used (the same as on ISE/ACS).

Setting this value is a valid workaround in order to fix issues in topology like this:

Supplicant [MTU 1500] ---- ---- [MTU 9000]Switch[MTU 9000] ----- ----- [MTU 9000]NPS

Currently switches do not allow you to set the MTU per port; for 6880 switches, this feature is added with

Cisco bug ID CSCuo26327 - 802.1x EAP-TLS not working on FEX host ports.

AnyConnect

AnyConnect sends EAP-TLS fragments (usually Client Certificate) that are 1,486 bytes long. For this value size, the Ethernet frame is 1,500 bytes. The last fragment is usually smaller.

Microsoft Windows Native Supplicant

Microsoft Windows sends EAP-TLS fragments (usually Client Certificate) that are 1,486 or 1,482 bytes long. For this value size, the Ethernet frame is 1,500 bytes. The last fragment is usually smaller.

Related Information

- **Configuring IEEE 802.1x Port-Based Authentication**
- **Technical Support & Documentation - Cisco Systems**