RADIUS EXTensions Working Group
Internet-Draft
Intended status: Experimental
Expires: July 5, 2013

A. Perez-Mendez
R. Marin-Lopez
F. Pereniguez-Garcia
G. Lopez-Millan
University of Murcia
D. Lopez
Telefonica I+D
A. DeKok
Network RADIUS
Jan 2013

Support of fragmentation of RADIUS packets
draft-perez-radext-radius-fragmentation-04

Abstract

This document describes a mechanism providing fragmentation support of RADIUS packets that exceed the 4 KB limit.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 5, 2013.

Table of Contents

1. Introduction

RADIUS [RFC2865] is a protocol for carrying authentication, authorization, and configuration information between a Network Access Server (NAS) which desires to authenticate its links and a shared Authentication Server (AS). Information is exchanged between the NAS and the AS through RADIUS packets. Each RADIUS packet can transport several RADIUS attributes, to convey the necessary information to the other peer, up to a maximum size of 4 KB of total data (including RADIUS packet headers). RADIUS attributes have a maximum payload size of 253 bytes.

RADIUS has been extensively used for the years. Along this time, the need of sending RADIUS attributes larger than 253 bytes has become a reality. An immediate alternative to overcome this issue consists in truncating the data into a group of RADIUS attributes of the same type, and then insert them ordered into the RADIUS packet. At the destination, the content of these attributes is extracted and joined to rebuild the original data. This scheme is followed, for example, by RADIUS-EAP [RFC3579]. A more advanced solution is given in [I-D.ietf-radext-radius-extensions], where extended attributes can be marked with a flag to indicate fragmentation. A reference-based mechanism is also proposed in [RFC6158], where attributes can be obtained through an out-of-band protocol.

However, there are no proposals to deal with fragmentation at a packet level, when the total size exceeds the 4 KB limit imposed by the RADIUS specification. When RADIUS is considered in more complex AAA scenarios, including the exchange of bigger amount of data, like SAML assertions or JSON Web tokens, exceeding this limit becomes more likely, thus making necessary the availability of mechanisms for dealing with this situation.

This document defines a mechanism to allow RADIUS peers to exchange packets exceeding the 4 KB limit, by fragmenting them across several exchanges. This proposal tries to maintain compatibility with intra-packet fragmentation mechanisms (like those defined in [RFC3579] or in [I-D.ietf-radext-radius-extensions]) and with the existing RADIUS deployments.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Overview

When a RADIUS client or server need to send a packet that exceeds 4 KB, the mechanism proposed in this document is used. First, the large packet is truncated into several smaller RADIUS packets (i.e. chunks) of the same type (e.g. Access-Request). Each small RADIUS packet is so-called "chunk" in this specification. The first chunk contains the first "n" RADIUS attributes of the original packet (in the same order), until a limit below 4096 bytes. The actual amount of data from the original packet included into each chunk will depend on the specific length of the attributes, the amount of proxies between both ends, and the number of signaling attributes (more details in Section 4). If there are still attributes from the original packet that have not been yet included into any chunk, a new attribute called More-Data-Pending is appended into the chunk.

Then the first chunk is sent to the other party, which identifies the packet as a chunk (the More-Data-Pending attribute is present), and requests for the next chunk. The RADIUS State attribute and the RADIUS Identifier field are used to tie the conversation together.

This process is repeated until all the RADIUS attributes from the original packet have been sent by means of several chunks. Once all the chunks have been received by the peer, the original packet is reconstructed and processed as if received in one piece.

When a packet is truncated into chunks, a special situation may occur when it is combined with Extended Type attributes as defined in [I-D.ietf-radext-radius-extensions]. If the truncation splits an existing fragmented attribute along two or more chunks, the last fragment of that attribute for the first chunks will have the flag M enabled (indicating the attribute is not completed). This situation is specifically forbidden in [I-D.ietf-radext-radius-extensions]. To indicate that this situation is provoked by a truncation and hence MUST be allowed, a new flag "T" (indicating truncation) MUST be set into that Extended-Type-Flag attribute. The combination of the flags "M" and "T" indicates that the attribute is fragmented (flag M), but that all the fragments are not available in this chunk (flag T).

Indeed, this last situation will be the most usual. Typically, packet fragmentation will occur as a consequence of including one or more large (and fragmented) attributes into a RADIUS packet. Hence, the truncation will probably split the large attribute into two (or more) pieces. The rest of possibilities, where the truncation point does not split a fragmented attribute, do not require any special treatment.

Packet fragmentation may occur at any moment during a RADIUS

exchange, as peers may require to send a big amount of data at any moment. It is worth noting that certain RADIUS extensions already avoid that RADIUS packet exceed the 4 KB limit. This is the case of RADIUS-EAP [RFC3579]. Thus, it is envisaged that most of the times large packets will be generated by authorization data, which is sent along with the Access-Accept packet (e.g. SAML assertions, JWT, filters...).

3. Fragmentation of packets

3.1. Access-Request

When the NAS desires to send a RADIUS packet that exceeds the 4 KB limit, the packet can be split into smaller packets (chunks) and sent over different exchanges. This fact is indicated by including a More-Data-Pending attribute on each chunk (except the last one of the series). The process is described in detail using the following example. In this example, attributes "Data" and "Other" are Extended-Type-Flags, as defined in [I-D.ietf-radext-radius-extensions]. All the packets belonging to the same fragmentation exchange are tied together by making use of the standard RADIUS mechanisms, that is, the use of the RADIUS State attribute by the Server, and the use of the Identifier field of the RADIUS packet by the Client.

In order to make the example simpler, it is assumed that each RADIUS packet can include up to 8 RADIUS attributes, instead of using bytes. Flag M is indicated as [M]. Flag T is indicated as [T]. Presence of both is indicated as [MT]. Data(1), Data(2), Data(3)... indicate successive fragments of the attribute "Data", while Other(1), Other(2) and Other(3) indicate successive fragments of the attribute "Other". The Identifier field is denoted as IDX, even though it is not an attribute.

- The RADIUS client wants to send the following RADIUS packet:

  Access-Request = ID1, User-Name, Calling-Station-Id, Data(1)[M], Data(2)[M], Data(3)[M], Data(4)[M], Data(5)[M], Data(6)[M], Data(7)[M], Data(8)[M], Data(9), Other(1)[M], Other(2)[M], Other(3)

- As the RADIUS packet exceeds the maximum allowed length (8 attributes), the RADIUS client truncates the packet to generate the first chunk, including the More-Data-Pending attribute. As attribute "Data" does not completely fit into this chunk, the flag "T" is activated into the last fragment included into this chunk (i.e. Data(5)), to indicate that it is not the last fragment of

the attribute. If the original Access-Request packet contains a User-Name attribute, it MUST be included on every chunk sent to the server. This is required because proxies may need this value to forward the chunk to its proper destination.

Access-Request-1 = ID1, User-Name, Calling-Station-Id, Data(1)[M], Data(2)[M], Data(3)[M], Data(4)[M], Data(5)[MT], More-Data-Pending

- When the server receives the RADIUS packet containing the More-Data-Pending attribute, the processing of the packet is delayed until all the pending data is received. The pending data is requested by means of an Access-Challenge packet, using the State attribute to tie together this response with the subsequent request from the client.

  Access-Challenge-1 = ID1, State1

- The client continues including attributes until another chunk is completed, appending again the More-Data-Pending attribute. The State attribute received in the Access-Challenge (State1) is also included in this chunk. As attribute "Other" does not completely fit into this chunk, the flag "T" is activated into the last fragment included into this chunk (i.e. Other(1)), to indicate that it is not the last fragment of the attribute.

  Acess-Request-2 = ID2, User-Name, State1, Data(6)[M], Data(7)[M], Data(8)[M], Data(9), Other(1)[MT], More-Data-Pending

- As the received request contains the More-Data-Pending attribute, the server stores the attributes into the state associated to State1 and replies with another Access-Challenge. The challenge contains a new State attribute (State2) that refers to this conversation.

  Access-Challenge-2 = ID2, State2

- Finally, the client sends the last chunk of the original packet, including the received State attribute (State2).

  Access-Request-3 = ID3, User-Name, State2, Other(2)[M], Other(3)

- On reception of this last chunk (no More-Data-Pending attribute present), the server can process the received attributes as if they all had been received into a single RADIUS packet larger than 4 KB. See section Section 5.4 for further details.

The Figure 1 depicts this scenario.

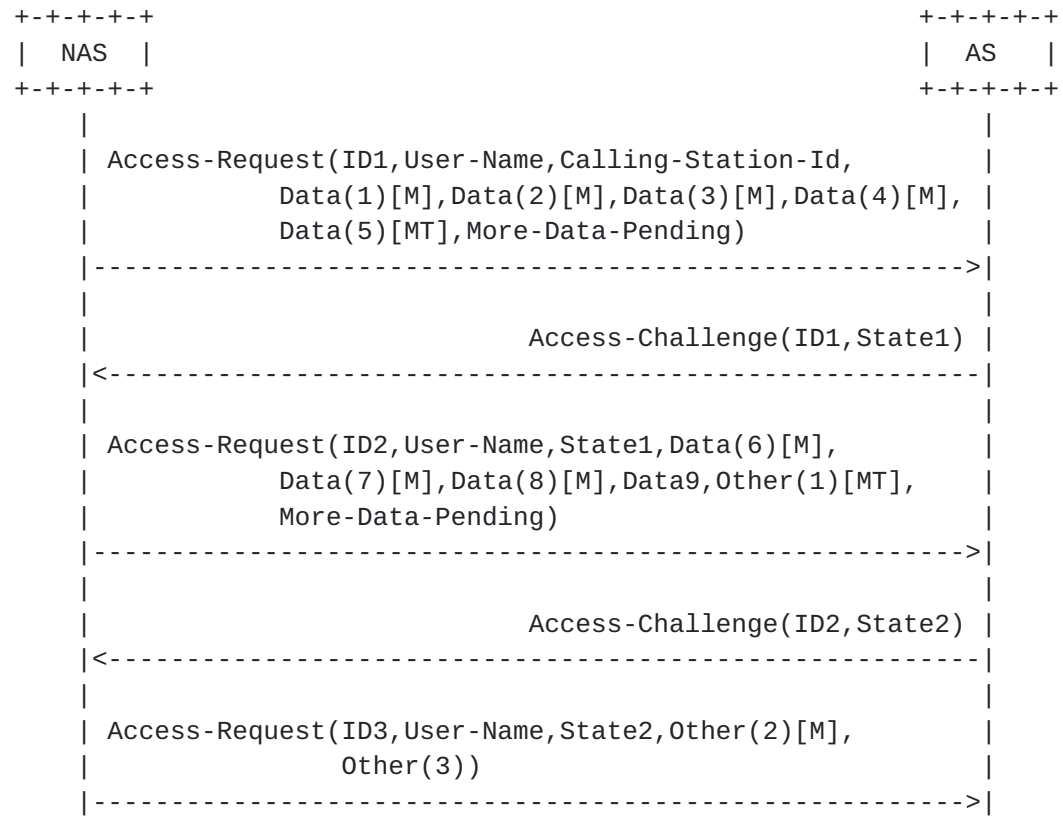


Figure 1: Fragmented Access-Request packet

3.2. Access-Challenge

When the server (AS) wants to send a large RADIUS Access-Challenge packet, the solution is very similar to the previous one. The difference in this scenario is that the AS includes a RADIUS State attribute along with the More-Data-Required.

If the Access-Request packet that motivated the generation of the fragmented Access-Challenge contained a User-Name attribute, the RADIUS client MUST include this attribute on every Access-Request packet it sends to request more chunks. This is required to allow proxies to determine where to forward packets.

o The RADIUS server wants to send the following RADIUS packet:

Access-Challenge = ID1, Data(1)[M], Data(2)[M], Data(3)[M], Data(4)[M], Data(5)[M], Data(6)[M], Data(7)[M], Data(8)[M], Data(9)[M], Data(10), Other(1)[M], Other(2)[M], Other(3)

This packet is a response for an Access-Request packet sent by the

client with Identifier=ID1.

- As the RADIUS packet exceeds the maximum allowed length (8 attributes), the RADIUS server truncates the packet to generate the first chunk, including the More-Data-Pending attribute and a RADIUS State attribute to allow recognize the fragmentation conversation. As attribute "Data" does not completely fit into this chunk, the flag "T" is activated into the last fragment included into this chunk (i.e. Data(6)), to indicate that it is not the last fragment of the attribute.

  Access-Challenge-1 = ID1, Data(1)[M], Data(2)[M], Data(3)[M], Data(4)[M], Data(5)[M], Data(6)[MT], More-Data-Pending, State1

- When the RADIUS client receives the RADIUS packet containing the More-Data-Pending attribute, the processing of the packet is delayed until all the pending data is received. The pending data is requested by means of an Access-Request packet, including the received State attribute to tie together this Access-Request with the previous Access-Challenge sent by the server. A new ID is generated for this request.

  Access-Request-2 = ID2, User-Name, State1

- The server continues including attributes until another chunk is completed, appending again a More-Data-Pending attribute and a new RADIUS State attribute.

  Access-Challenge-2 = ID2, Data(7)[M], Data(8)[M], Data(9)[M], Data(10), Other(1)[M], Other(2)[MT], More-Data-Pending, State2

- The client recognizes the received chunk as a continuation of the fragmentation conversation by means of the ID field. As the More-Data-Pending attribute is present, the client stores the rest of attributes into the state associated to ID2 and replies with another Access-Request to request more data. The Access-Request contains a new ID that refers to this conversation.

  Access-Request-3 = ID3, User-Name, State2

- Finally, the server sends the last chunk of the original packet. This chunk does not include any More-Data-Pending nor RADIUS State attribute (unless the original large packet contained a RADIUS State attribute, see section Section 5.2.

  Access-Challenge-3 = ID3, Other(3)

o On reception of this last chunk, the client can process the received attributes as if they all had been received into a single RADIUS packet larger than 4 KB. See section Section 5.4 for further details.

The Figure 2 depicts how the message exchange would be if the AS wants to send a large packet to the NAS.

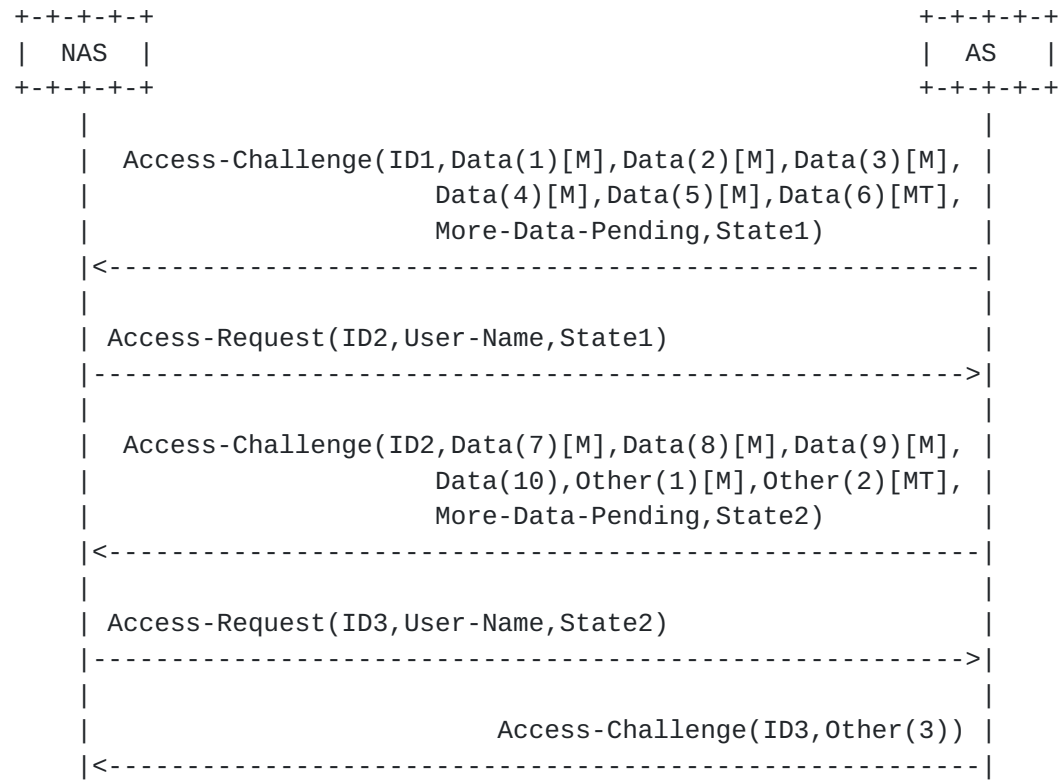


Figure 2: Fragmented Access-Challenge packet

3.3. Access-Accept

If the AS wants to send an Access-Accept packet that exceeds the 4 KB limit (e.g. due to the inclusion of authorization-specific attributes), the operation is slightly different. As some attributes are allowed to appear in Access-Accept packets, but they cannot be present in Access-Challenge packets, the solution described in the previous sections is not directly applicable. Instead, the AS MUST start by sending an Access-Accept packet to the NAS, containing all the attributes that can not be included in Access-Challenge packets, and including a Service-Type attribute with value "Additional-Authorization". This is a new value indicating that, though the end user has been successfully authenticated, additional authorization information needs to be retrieved before access can be completely granted. Existing values of Service-Type cannot be used as they are

defined for other purposes. If the attributes that can not be included in RADIUS Access-Challenge packets exceed the limit of the Access-Accept packet, an error will be generated as the server is not able to send them to the client.

The rest of the attributes are received via a series of Access-Request/Access-Challenge exchanges, as described in the previous section. Finally, the last chunk sent from the AS to the NAS would be an Access-Accept containing the last attributes of the original packet. If the original large Access-Accept packet contained a Service-Type attribute, it will be included in this last chunk to avoid confusion with the one used for signaling in the first chunk.

The following example depicts the fragmentation of an Access-Accept packet. For simplicity, in the example, Service-Type[X] indicates a Service-Type attribute of value X.

- The AS wants to send the following Access-Accept packet:

  Access-Accept = ID1, User-Name, Service-Type[X], Framed-IP-Address, Data(1)M], Data(2)[M], Data(3)[M], Data(4)[M], Data(5)[M], Data(6)[M], Data(7)[M], Data(8)[M], Data(9)[M], Data(10)

- As the RADIUS packet exceeds the maximum allowed length (8 attributes), the AS truncates the packet to generate the first chunk. This chunk only includes the attributes that cannot be included in Access-Challenge packets. In this example they are User-Name, Service-Type (indicating additional authorization) and Framed-IP-Address. Note that the Service-Type attribute is changed from "X" to "Additional-Authorization", and a new State attribute is included.

  Access-Accept-1 = ID1, User-Name, Service-Type[Additional-Authorization], Framed-IP-Address, State1

- When the NAS receives the Access-Accept, it determines, based on the Service-Type=Additional-Authorization, that additional exchanges are required. Thus, it generates a new Access-Request packet containing the User-Name attribute along with the received State attribute.

  Access-Request-1 = ID2, User-Name, State1

- The AS then generates a new chunk with part of the remaining attributes to be sent. As they do not fit into a single chunk, a More-Data-Pending attribute and a new State attribute are also included.

Access-Challenge-1 = ID2, Data(1)[M], Data(2)[M], Data(3)[M], Data(4)[M], Data(5)[M], Data(6)[MT], More-Data-Pending, State2

- The NAS determines the received packet is part of a larger one (i.e. it is a chunk) due to the presence of the More-Data-Pending attribute, hence it requests the rest of the data by sending a new Access-Request packet including the received State attribute.

  Access-Request-2 = ID3, User-Name, State2

- Finally, the AS includes the rest of the attributes into the final Access-Accept packet. This packet also includes the original Service-Type for the user.

  Access-Accept-2 = ID3, Data(7)[M], Data(8)[M], Data(9)[M], Data(10), Service-Type[X]

- On reception of this last packet, the NAS can process the totality of the received attributes as if they were all received into a single RADIUS packet larger than 4 KB. See section Section 5.4 for further details on this.

The following figure depicts the exchange of chunks between the NAS and the AS.

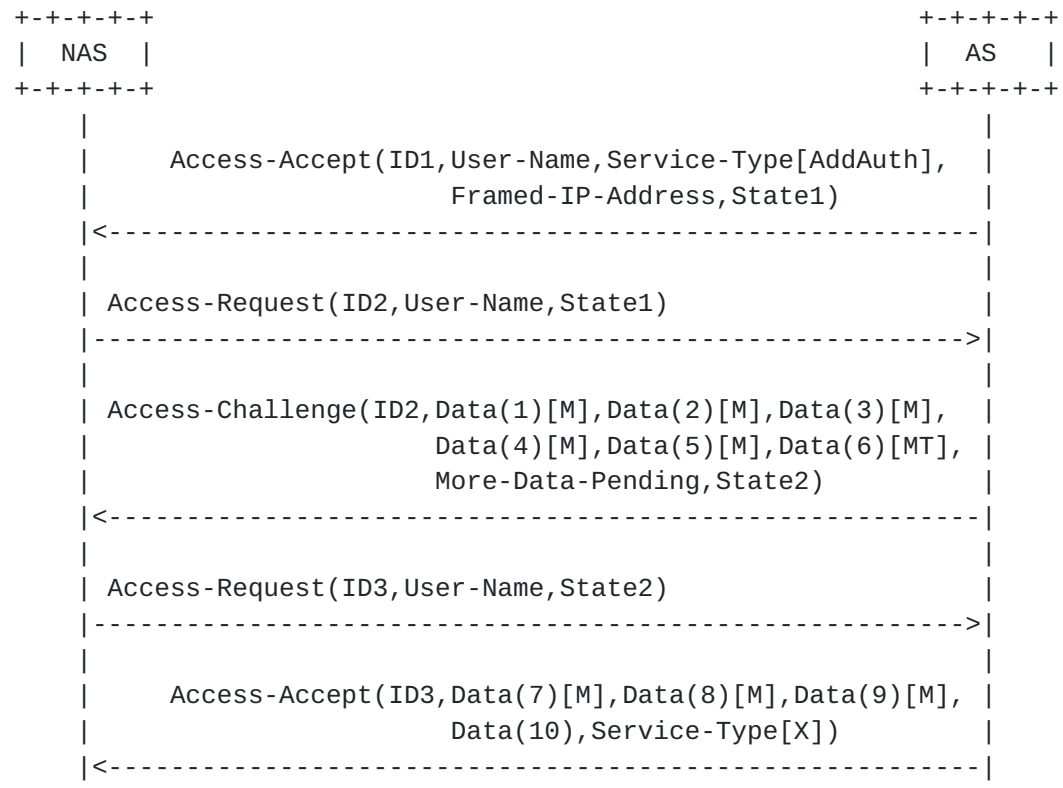


Figure 3: Fragmented Access-Accept packet

4. Chunk size

In an ideal scenario, chunks would be exactly 4096 bytes length, and they would contain exactly 4096-20=4076 bytes of attributes from the original large packet (where 20 is the size of he RADIUS header). In this way, the number of round trips required to send a large packet would be optimal. However, this is not possible for several reasons.

1. RADIUS attributes have a variable length, and must be included completely in a chunk. Thus, it is possible that, even if there is some free space in the chunk, it is not enough to include the next attribute. This can generate up to 254 bytes of spare space on every chunk.

2. RADIUS fragmentation requires the introduction of some extra attributes for signaling. Specifically, a More-Data-Pending (4 bytes length) attribute is included on every chunk of a packet, except the last one. A RADIUS State attribute (from 3 to 255 bytes) is also included in most chunks, to allow the server to bind an Access-Request with a previous Access-Challenge. User-Name attributes (from 3 to 255 bytes) are introduced on every chunk the client sends as they are required by the proxies to route the packet to its destination. Together, these attributes can generate from up to 10 to 514 bytes of signaling data, reducing the amount of payload information that can be sent on each chunk.

3. RADIUS packets SHOULD be adjusted to avoid exceeding the network MTU. Otherwise, IP fragmentation may occur, having undesirable consequences. Hence, maximum chunk size would be decreased from 4096 to the actual MTU of the network.

4. The inclusion of Proxy-State attributes by intermediary proxies can decrease the availability of usable space into the chunk. This is described with further detail in Section 5.1.

5. Handling special attributes

5.1. Proxy-State attribute

RADIUS proxies may introduce Proxy-State attributes into any Access-Request packet they forward. Should they cannot add this information to the packet, they may silently discard forwarding it to its destination, leading to DoS situations. Moreover, any Proxy-State

attribute received by a RADIUS server in an Access-Request packet MUST be copied into the reply packet to it. For these reasons, Proxy-State attributes require a special treatment within the packet fragmentation mechanism.

When the RADIUS server replies to an Access-Request packet as part of a conversation involving a fragmentation (either a chunk or a request for chunks), it MUST include every Proxy-State attribute received into the reply packet. This means that the server MUST take into account the size of these Proxy-State attributes in order to calculate the size of the next chunk to be sent.

However, while a RADIUS server will always know how many space MUST be left on each reply packet for Proxy-State attributes (as they are directly included by the RADIUS server), a RADIUS client cannot know this information, as Proxy-State attributes are removed from the reply packet by their respective proxies before forwarding them back. Hence, clients need a mechanism to discover the amount of space required by proxies to introduce their Proxy-State attributes. In the following we describe a new mechanism to perform such a discovery:

1. When a RADIUS client does not know how many space will be required by intermediate proxies for including their Proxy-State attributes, it SHOULD start using a conservative value (e.g. 1 KB) as the chunk size.

2. When the RADIUS server receives a chunk from the client, it can calculate the total size of the Proxy-State attributes that have been introduced by intermediary proxies along the path. This information MUST be returned to the client in the next reply packet, encoded into a new attribute called Proxy-State-Len.

3. The RADIUS client reacts upon the reception of this attribute by adjusting the maximum size for the next chunk accordingly.

5.2. State attribute

This RADIUS fragmentation mechanism makes use of the State attribute to link all the chunks belonging to the same fragmented packet. However, some considerations are required when the RADIUS server is fragmenting a packet that already contains a State attribute for other purposes not related with the fragmentation. If the procedure described in Section 3 is followed, two different State attributes could be included into a single chunk, incurring into two problems. First, [RFC2865] explicitly forbids that more than one State attribute appears into a single Access-Challenge packet.

A straightforward solution consists on making the RADIUS server to send the original State attribute into the last chunk of the sequence (attributes can be re-ordered as specified in [RFC2865]). As the last chunk (when generated by the RADIUS server) does not contain any State attribute due to the fragmentation mechanism, both situations described above are avoided.

Something similar happens when the RADIUS client has to send a fragmented packet that contains a State attribute on it. The client MUST assure that this original State is included into the first chunk sent to the server (as this one never contains any State attribute due to fragmentation).

5.3. Interaction with RADIUS-EAP

When the fragmented packet belongs to a RADIUS-EAP [RFC3579] conversation, the EAP-Message, Message-Authenticator, and State attributes MUST be sent in the same chunk. This can be achieved by moving them to the last chunk in the case of the server, or to the first chunk in the case of the client.

5.4. Rebuilding the original large packet

The RADIUS client stores the RADIUS attributes received on each chunk in order to be able to rebuild the original large packet after receiving the last chunk. However, some of these received attributes MUST not be stored in this list, as they have been introduced as part of the fragmentation signaling and hence, they are not part of the original packet.

- State (except the one in the last chunk, if present)
- Service-Type=Additional-Authorization
- More-Data-Pending
- Proxy-State-Len

Similarly, the RADIUS server MUST NOT store the following attributes as part of the original large packet:

- State (except the one in the first chunk, if present)
- More-Data-Pending
- Proxy-State (except the ones in the last chunk)

o User-Name (except the one in the first chunk)

6. New attribute definition

This document proposes the definition of two new extended type attributes, called More-Data-Pending and Proxy-State-Len. The format of these attributes follows the indications for an Extended Type attribute defined in [I-D.ietf-radext-radius-extensions].

6.1. More-Data-Pending attribute

This attribute indicates that a RADIUS packet is not complete (that is, it is a chunk), and more chunks MUST be received to regenerate the original packet. The following figure represents the format of the More-Data-Pending attribute.

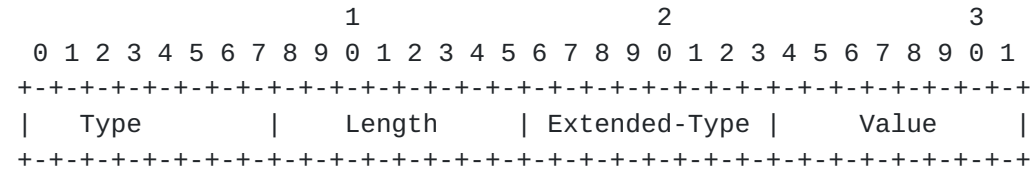


Figure 4: More-Data-Pending format

Type

To be assigned (TBA)

Length

4

Extended-Type

To be assigned (TBA).

Value

1 byte. This field is reserved for future use. It MUST be set to zero and ignored by the receiver.

This attribute MAY be present in Access-Request and Access-Challenge packets. It MUST not be included in Access-Accept packets (see section Section 3.3.

6.2. Proxy-State-Len attribute

This attribute indicates to the RADIUS client the length of the Proxy-State attributes received by the RADIUS server. This information is useful to adjust the length of the chunks sent by the RADIUS client. The format of this Proxy-State-Len attribute is the following:

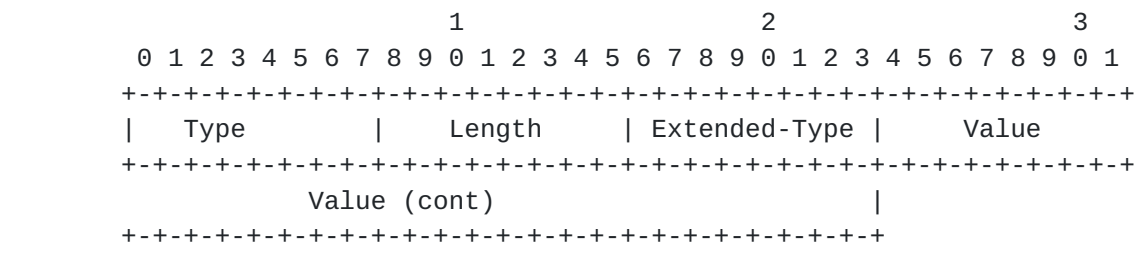


Figure 5: Proxy-State-Len format

Type

To be assigned (TBA)

Length

7

Extended-Type

To be assigned (TBA).

Value

4 bytes. Total length (in bytes) of received Proxy-State attributes (including headers).

This attribute MAY be present in Access-Challenge packets. It MUST not be included in Access-Accept packets.

6.3. Table of attributes

The following table shows the different attributes defined in this document related with the kind of RADIUS packets where they can be present.

	Kind of packet			
Attribute Name	Req	Acc	Rej	Cha
More-Data-Pending	0-1	0	0	0-1
Proxy-State-Len	0	0-1	0	0-1

Figure 6

7. Operation with proxies

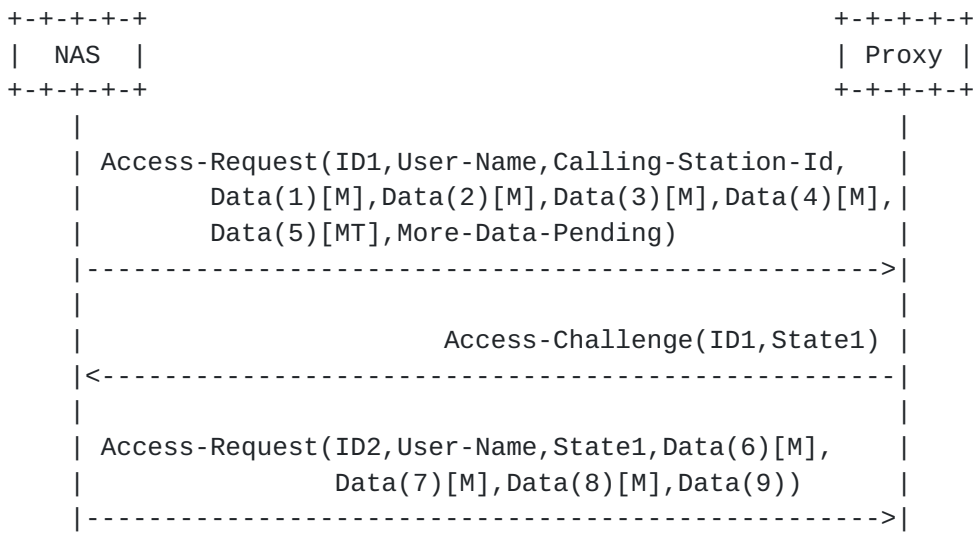
The fragmentation mechanism defined above is designed to be transparent to legacy proxies, as long as they do not want to modify any fragmented attribute. Nevertheless, updated proxies supporting this specification can even modify fragmented attributes.

7.1. Legacy proxies

As every chunk is indeed a RADIUS packet, legacy proxies treat them as the rest of packets, routing them to their destination. Proxies can introduce Proxy-State attributes to Access-Request packets, even if they are indeed chunks. This will not affect how fragmentation is managed. The server will include all the received Proxy-State attributes into the generated response, as described in [RFC2865]. Hence, proxies do not distinguish between a regular RADIUS packet and a chunk.

7.2. Updated proxies

Updated proxies can interact with clients and servers in order to obtain the complete large packet before start forwarding it. In this way, proxies can manipulate (modify and/or remove) any attribute of the packet, or introduce new attributes, without worrying about crossing the boundaries of the chunk size. Once the manipulated packet is ready, it is sent to the original destination using the fragmentation mechanism (if required). The following example shows how an updated proxy interacts with the NAS to obtain a large Access-Request packet, modify an attribute resulting into a even more large packet, and interacts with the AS to complete the transmission of the modified packet.



PROXY MODIFIES ATTRIBUTE Data INCREASING ITS
SIZE FROM 9 FRAGMENTS TO 11 FRAGMENTS

Figure 7: Updated proxy interacts with NAS

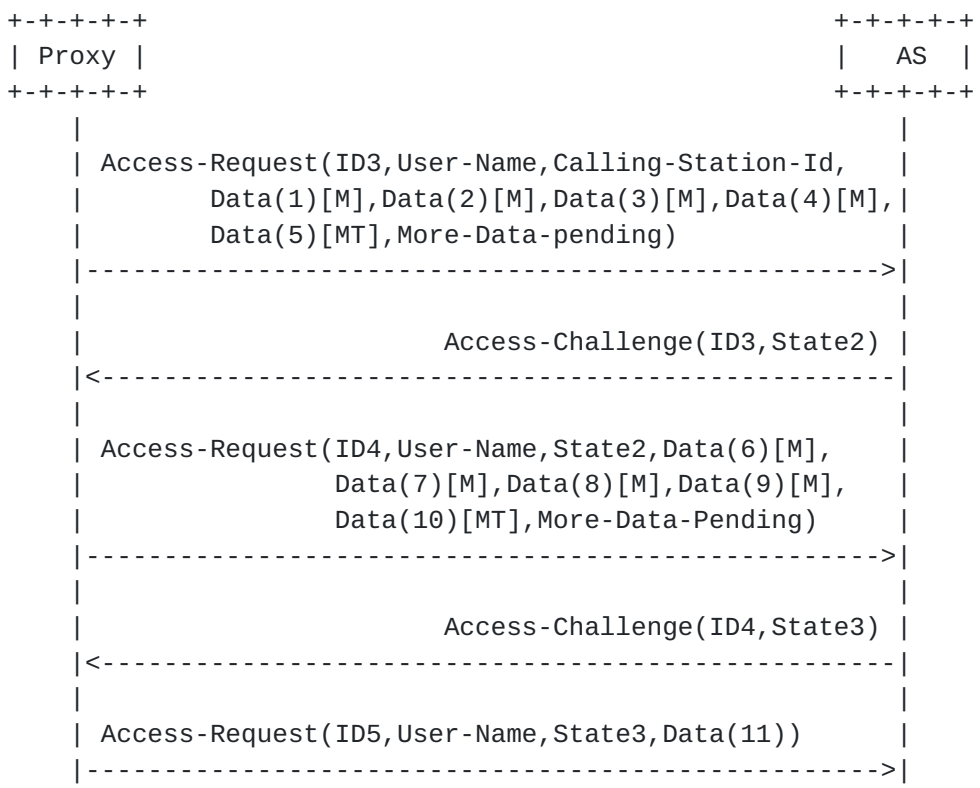


Figure 8: Updated proxy interacts with AS

8. Security Considerations

Proxies can modify chunks in such a way that the fragmentation process fails. Nevertheless, RADIUS proxies are trusted entities, and they are always allowed to modify packets completely.

9. IANA Considerations

This document has no actions for IANA.

10. Normative References

[I-D.ietf-radext-radius-extensions]
DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", draft-ietf-radext-radius-extensions-08 (work in progress), January 2013.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.

[RFC6158] DeKok, A. and G. Weber, "RADIUS Design Guidelines", BCP 158, RFC 6158, March 2011.

Authors' Addresses

Alejandro Perez-Mendez (Ed.)
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 46 44
Email: alex@um.es

Rafa Marin-Lopez
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 85 01
Email: rafa@um.es

Fernando Pereniguez-Garcia
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 78 82
Email: pereniguez@um.es

Gabriel Lopez-Millan
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 85 04
Email: gabilm@um.es

Diego R. Lopez
Telefonica I+D
Don Ramon de la Cruz, 84
Madrid, 28006
Spain

Phone: +34 913 129 041
Email: diego@tid.es

Alan DeKok
Network RADIUS
15 av du Granier
Meylan, 38240
France

Phone: +34 913 129 041
Email: aland@networkradius.com
URI: http://networkradius.com