Internet Engineering Task Force (IETF)
Request for Comments: 7499
Category: Experimental
ISSN: 2070-1721

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

April 2015

Support of Fragmentation of RADIUS Packets

Abstract

The Remote Authentication Dial-In User Service (RADIUS) protocol is limited to a total packet size of 4096 bytes. Provisions exist for fragmenting large amounts of authentication data across multiple packets, via Access-Challenge packets. No similar provisions exist for fragmenting large amounts of authorization data. This document specifies how existing RADIUS mechanisms can be leveraged to provide that functionality. These mechanisms are largely compatible with existing implementations, and they are designed to be invisible to proxies and "fail-safe" to legacy RADIUS Clients and Servers.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7499.

Table of Contents

1. Introduction 4
1.1. Requirements Language 6
2. Status of This Document 6
3. Scope of This Document 7
4. Overview 10
5. Fragmentation of Packets 13
5.1. Pre-Authorization 14
5.2. Post-Authorization 18
6. Chunk Size 21
7. Allowed Large Packet Size 22
8. Handling Special Attributes 23
8.1. Proxy-State Attribute 23
8.2. State Attribute 24
8.3. Service-Type Attribute 25
8.4. Rebuilding the Original Large Packet 25
9. New T Flag for the Long Extended Type Attribute Definition 26
10. New Attribute Definition 26
10.1. Frag-Status Attribute 27
10.2. Proxy-State-Length Attribute 28
10.3. Table of Attributes 29
11. Operation with Proxies 29
11.1. Legacy Proxies 29
11.2. Updated Proxies 29
12. General Considerations 31
12.1. T Flag 31
12.2. Violation of RFC 2865 32
12.3. Proxying Based on User-Name 32
12.4. Transport Behavior 33
13. Security Considerations 33
14. IANA Considerations 34
15. References 35
15.1. Normative References 35
15.2. Informative References 35
Acknowledgements 37
Authors' Addresses 37

1. Introduction

The RADIUS [RFC2865] protocol carries authentication, authorization, and accounting information between a RADIUS Client and a RADIUS Server. Information is exchanged between them through RADIUS packets. Each RADIUS packet is composed of a header, and zero or more attributes, up to a maximum packet size of 4096 bytes. The protocol is a request/response protocol, as described in the operational model ([RFC6158], Section 3.1).

The intention of the above packet size limitation was to avoid UDP fragmentation as much as possible. Back then, a size of 4096 bytes seemed large enough for any purpose. Now, new scenarios are emerging that require the exchange of authorization information exceeding this 4096-byte limit. For instance, the Application Bridging for Federated Access Beyond web (ABFAB) IETF working group defines the transport of Security Assertion Markup Language (SAML) statements from the RADIUS Server to the RADIUS Client [SAML-RADIUS]. This assertion is likely to be larger than 4096 bytes.

This means that peers desiring to send large amounts of data must fragment it across multiple packets. For example, RADIUS-EAP [RFC3579] defines how an Extensible Authentication Protocol (EAP) exchange occurs across multiple Access-Request / Access-Challenge sequences. No such exchange is possible for accounting or authorization data. [RFC6158], Section 3.1 suggests that exchanging large amounts of authorization data is unnecessary in RADIUS. Instead, the data should be referenced by name. This requirement allows large policies to be pre-provisioned and then referenced in an Access-Accept. In some cases, however, the authorization data sent by the RADIUS Server is large and highly dynamic. In other cases, the RADIUS Client needs to send large amounts of authorization data to the RADIUS Server. Neither of these cases is met by the requirements in [RFC6158]. As noted in that document, the practical limit on RADIUS packet sizes is governed by the Path MTU (PMTU), which may be significantly smaller than 4096 bytes. The combination of the two limitations means that there is a pressing need for a method to send large amounts of authorization data between RADIUS Client and Server, with no accompanying solution.

[RFC6158], Section 3.1 recommends three approaches for the transmission of large amounts of data within RADIUS. However, they are not applicable to the problem statement of this document for the following reasons:

- The first approach (utilization of a sequence of packets) does not talk about large amounts of data sent from the RADIUS Client to a RADIUS Server. Leveraging EAP (request/challenge) to send the data is not feasible, as EAP already fills packets to PMTU, and not all authentications use EAP. Moreover, as noted for the NAS-Filter-Rule attribute ([RFC4849]), this approach does not entirely solve the problem of sending large amounts of data from a RADIUS Server to a RADIUS Client, as many current RADIUS attributes are not permitted in Access-Challenge packets.

- The second approach (utilization of names rather than values) is not usable either, as using names rather than values is difficult when the nature of the data to be sent is highly dynamic (e.g., a SAML statement or NAS-Filter-Rule attributes). URLs could be used as a pointer to the location of the actual data, but their use would require them to be (a) dynamically created and modified, (b) securely accessed, and (c) accessible from remote systems. Satisfying these constraints would require the modification of several networking systems (e.g., firewalls and web servers). Furthermore, the setup of an additional trust infrastructure (e.g., Public Key Infrastructure (PKI)) would be required to allow secure retrieval of the information from the web server.

- PMTU discovery does not solve the problem, as it does not allow the sending of data larger than the minimum of (PMTU or 4096) bytes.

This document provides a mechanism to allow RADIUS peers to exchange large amounts of authorization data exceeding the 4096-byte limit by fragmenting it across several exchanges. The proposed solution does not impose any additional requirements to the RADIUS system administrators (e.g., need to modify firewall rules, set up web servers, configure routers, or modify any application server). It maintains compatibility with intra-packet fragmentation mechanisms (like those defined in [RFC3579] or [RFC6929]). It is also transparent to existing RADIUS proxies, which do not implement this specification. The only systems needing to implement this RFC are the ones that either generate or consume the fragmented data being transmitted. Intermediate proxies just pass the packets without changes. Nevertheless, if a proxy supports this specification, it may reassemble the data in order to examine and/or modify it.

A different approach to deal with RADIUS packets above the 4096-byte limit is described in [RADIUS-Larger-Pkts], which proposes to extend RADIUS over TCP by allowing the Length field in the RADIUS header to take values up to 65535 bytes. This provides a simpler operation, but it has the drawback of requiring every RADIUS proxy in the path between the RADIUS Client and the RADIUS Server to implement the extension as well.

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]. When these words appear in lower case, they have their natural language meaning.

2. Status of This Document

This document is an Experimental RFC. It defines a proposal to allow the sending and receiving of data exceeding the 4096-byte limit in RADIUS packets imposed by [RFC2865], without requiring the modification of intermediary proxies.

The experiment consists of verifying whether the approach is usable in a large-scale environment, by observing the uptake, usability, and operational behavior it shows in large-scale, real-life deployments. In that sense, so far the main use case for this specification is the transportation of large SAML statements defined within the ABFAB architecture [ABFAB-Arch]. Hence, it can be tested wherever an ABFAB deployment is being piloted.

Besides, this proposal defines some experimental features that will need to be tested and verified before the document can be considered for the Standards Track. The first one of them is the requirement of updating [RFC2865] in order to relax the sentence defined in Section 4.1 of that document that states that "An Access-Request MUST contain either a User-Password or a CHAP-Password or a State." This specification might generate Access-Request packets without any of these attributes. Although all known implementations have chosen the philosophy of "be liberal in what you accept," we need to gain more operational experience to verify that unmodified proxies do not drop these types of packets. More details on this aspect can be found in Section 12.2.

Another experimental feature of this specification is that it requires proxies to base their routing decisions on the value of the RADIUS User-Name attribute. Our experience is that this is the common behavior; thus, no issues are expected. However, it needs to be confirmed after using different implementations of intermediate proxies. More details on this aspect can be found in Section 12.3.

Moreover, this document requires two minor updates to Standards Track documents. First, it modifies the definition of the Reserved field of the Long Extended Type attribute [RFC6929] by allocating an additional flag called the T (Truncation) flag. No issues are expected with this update, although some proxies might drop packets that do not have the Reserved field set to 0. More details on this aspect can be found in Section 12.1.

The other Standards Track document that requires a minor update is [RFC6158]. It states that "attribute designers SHOULD NOT assume that a RADIUS implementation can successfully process RADIUS packets larger than 4096 bytes," something no longer true if this document advances.

A proper "Updates" clause will be included for these modifications when/if the experiment is successful and this document is reissued as a Standards Track document.

3. Scope of This Document

This specification describes how a RADIUS Client and a RADIUS Server can exchange data exceeding the 4096-byte limit imposed by one packet. However, the mechanism described in this specification SHOULD NOT be used to exchange more than 100 kilobytes of data. Any more than this may turn RADIUS into a generic transport protocol, such as TCP or the Stream Control Transmission Protocol (SCTP), which is undesirable. Experience shows that attempts to transport bulk data across the Internet with UDP will inevitably fail, unless these transport attempts reimplement all of the behavior of TCP. The underlying design of RADIUS lacks the proper retransmission policies or congestion control mechanisms that would make it a competitor of TCP.

Therefore, RADIUS/UDP transport is by design unable to transport bulk data. It is both undesirable and impossible to change the protocol at this point in time. This specification is intended to allow the transport of more than 4096 bytes of data through existing RADIUS/UDP proxies. Other solutions such as RADIUS/TCP MUST be used when a "green field" deployment requires the transport of bulk data.

Section 7, below, describes in further detail what is considered to be a reasonable amount of data and recommends that administrators adjust limitations on data transfer according to the specific capabilities of their existing systems in terms of memory and processing power.

Moreover, its scope is limited to the exchange of authorization data, as other exchanges do not require such a mechanism. In particular, authentication exchanges have already been defined to overcome this limitation (e.g., RADIUS-EAP). Moreover, as they represent the most critical part of a RADIUS conversation, it is preferable to not introduce into their operation any modification that may affect existing equipment.

There is no need to fragment accounting packets either. While the accounting process can send large amounts of data, that data is typically composed of many small updates. That is, there is no demonstrated need to send indivisible blocks of more than 4 kilobytes of data. The need to send large amounts of data per user session often originates from the need for flow-based accounting. In this use case, the RADIUS Client may send accounting data for many thousands of flows, where all those flows are tied to one user session. The existing Acct-Multi-Session-Id attribute defined in [RFC2866], Section 5.11 has been proven to work here.

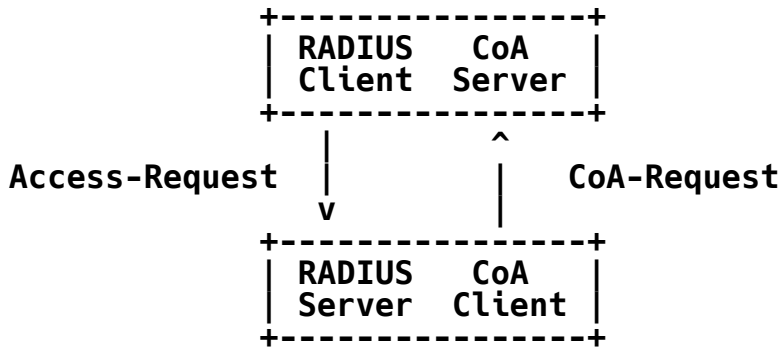
Similarly, there is no need to fragment Change-of-Authorization (CoA) [RFC5176] packets. Instead, according to [RFC5176], the CoA client will send a CoA-Request packet containing session identification attributes, along with Service-Type = Additional-Authorization, and a State attribute. Implementations not supporting fragmentation will respond with a CoA-NAK and an Error-Cause of Unsupported-Service.

The above requirement does not assume that the CoA client and the RADIUS Server are co-located. They may, in fact, be run on separate parts of the infrastructure, or even by separate administrators. There is, however, a requirement that the two communicate. We can see that the CoA client needs to send session identification attributes in order to send CoA packets. These attributes cannot be known a priori by the CoA client and can only come from the RADIUS Server. Therefore, even when the two systems are not co-located, they must be able to communicate in order to operate in unison. The alternative is for the two systems to have differing views of the users' authorization parameters; such a scenario would be a security disaster.

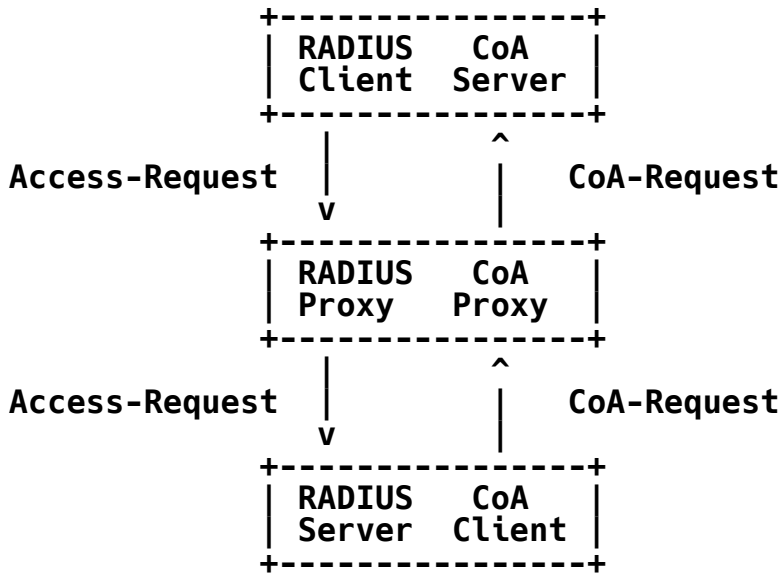
This specification does not allow for fragmentation of CoA packets. Allowing for fragmented CoA packets would involve changing multiple parts of the RADIUS protocol; such changes introduce the risk of implementation issues, mistakes, etc.

Where CoA clients (i.e., RADIUS Servers) need to send large amounts of authorization data to a CoA server (i.e., RADIUS Client), they need only send a minimal CoA-Request packet containing a Service-Type of Authorize Only, as per [RFC5176], along with session identification attributes. This CoA packet serves as a signal to the RADIUS Client that the users' session requires re-authorization. When the RADIUS Client re-authorizes the user via Access-Request, the RADIUS Server can perform fragmentation and send large amounts of authorization data to the RADIUS Client.

The assumption in the above scenario is that the CoA client and RADIUS Server are co-located, or at least strongly coupled. That is, the path from CoA client to CoA server SHOULD be the exact reverse of the path from RADIUS Client to RADIUS Server. The following diagram will hopefully clarify the roles:



Where there is a proxy involved:



That is, the RADIUS and CoA subsystems at each hop are strongly connected. Where they are not strongly connected, it will be impossible to use CoA-Request packets to transport large amounts of authorization data.

This design is more complicated than allowing for fragmented CoA packets. However, the CoA client and the RADIUS Server must communicate even when not using this specification. We believe that standardizing that communication and using one method for exchange of large data are preferred to unspecified communication methods and multiple ways of achieving the same result. If we were to allow fragmentation of data over CoA packets, the size and complexity of this specification would increase significantly.

The above requirement solves a number of issues. It clearly separates session identification from authorization. Without this separation, it is difficult to both identify a session and change its authorization using the same attribute. It also ensures that the authorization process is the same for initial authentication and for CoA.

4. Overview

Authorization exchanges can occur either before or after end-user authentication has been completed. An authorization exchange before authentication allows a RADIUS Client to provide the RADIUS Server with information that MAY modify how the authentication process will

be performed (e.g., it may affect the selection of the EAP method). An authorization exchange after authentication allows the RADIUS Server to provide the RADIUS Client with information about the end user, the results of the authentication process, and/or obligations to be enforced. In this specification, we refer to "pre-authorization" as the exchange of authorization information before the end-user authentication has started (from the RADIUS Client to the RADIUS Server), whereas the term "post-authorization" is used to refer to an authorization exchange happening after this authentication process (from the RADIUS Server to the RADIUS Client).

In this specification, we refer to the "size limit" as the practical limit on RADIUS packet sizes. This limit is the minimum between 4096 bytes and the current PMTU. We define below a method that uses Access-Request and Access-Accept in order to exchange fragmented data. The RADIUS Client and Server exchange a series of Access-Request / Access-Accept packets, until such time as all of the fragmented data has been transported. Each packet contains a Frag-Status attribute, which lets the other party know if fragmentation is desired, ongoing, or finished. Each packet may also contain the fragmented data or may instead be an "ACK" to a previous fragment from the other party. Each Access-Request contains a User-Name attribute, allowing the packet to be proxied if necessary (see Section 11.1). Each Access-Request may also contain a State attribute, which serves to tie it to a previous Access-Accept. Each Access-Accept contains a State attribute, for use by the RADIUS Client in a later Access-Request. Each Access-Accept contains a Service-Type attribute with the "Additional-Authorization" value. This indicates that the service being provided is part of a fragmented exchange and that the Access-Accept should not be interpreted as providing network access to the end user.

When a RADIUS Client or RADIUS Server needs to send data that exceeds the size limit, the mechanism proposed in this document is used. Instead of encoding one large RADIUS packet, a series of smaller RADIUS packets of the same type are encoded. Each smaller packet is called a "chunk" in this specification, in order to distinguish it from traditional RADIUS packets. The encoding process is a simple linear walk over the attributes to be encoded. This walk preserves the order of the attributes of the same type, as required by [RFC2865]. The number of attributes encoded in a particular chunk depends on the size limit, the size of each attribute, the number of proxies between the RADIUS Client and RADIUS Server, and the overhead for fragmentation-signaling attributes. Specific details are given in Section 6. A new attribute called Frag-Status (Section 10.1) signals the fragmentation status.

After the first chunk is encoded, it is sent to the other party. The packet is identified as a chunk via the Frag-Status attribute. The other party then requests additional chunks, again using the Frag-Status attribute. This process is repeated until all the attributes have been sent from one party to the other. When all the chunks have been received, the original list of attributes is reconstructed and processed as if it had been received in one packet.

The reconstruction process is performed by simply appending all of the chunks together. Unlike IPv4 fragmentation, there is no Fragment Offset field. The chunks in this specification are explicitly ordered, as RADIUS is a lock-step protocol, as noted in Section 12.4. That is, chunk N+1 cannot be sent until all of the chunks up to and including N have been received and acknowledged.

When multiple chunks are sent, a special situation may occur for Long Extended Type attributes as defined in [RFC6929]. The fragmentation process may split a fragmented attribute across two or more chunks, which is not permitted by that specification. We address this issue by using the newly defined T flag in the Reserved field of the Long Extended Type attribute format (see Section 9 for further details on this flag).

This last situation is expected to be the most common occurrence in chunks. Typically, packet fragmentation will occur as a consequence of a desire to send one or more large (and therefore fragmented) attributes. The large attribute will likely be split into two or more pieces. Where chunking does not split a fragmented attribute, no special treatment is necessary.

The setting of the T flag is the only case where the chunking process affects the content of an attribute. Even then, the Value fields of all attributes remain unchanged. Any per-packet security attributes, such as Message-Authenticator, are calculated for each chunk independently. Neither integrity checks nor security checks are performed on the "original" packet.

Each RADIUS packet sent or received as part of the chunking process MUST be a valid packet, subject to all format and security requirements. This requirement ensures that a "transparent" proxy not implementing this specification can receive and send compliant packets. That is, a proxy that simply forwards packets without detailed examination or any modification will be able to proxy "chunks".

5. Fragmentation of Packets

When the RADIUS Client or the RADIUS Server desires to send a packet that exceeds the size limit, it is split into chunks and sent via multiple client/server exchanges. The exchange is indicated via the Frag-Status attribute, which has value More-Data-Pending for all but the last chunk of the series. The chunks are tied together via the State attribute.

The delivery of a large fragmented RADIUS packet with authorization data can happen before or after the end user has been authenticated by the RADIUS Server. We can distinguish two phases, which can be omitted if there is no authorization data to be sent:

1. Pre-authorization. In this phase, the RADIUS Client MAY send a large packet with authorization information to the RADIUS Server before the end user is authenticated. Only the RADIUS Client is allowed to send authorization data during this phase.

2. Post-authorization. In this phase, the RADIUS Server MAY send a large packet with authorization data to the RADIUS Client after the end user has been authenticated. Only the RADIUS Server is allowed to send authorization data during this phase.

The following subsections describe how to perform fragmentation for packets for these two phases. We give the packet type, along with a RADIUS Identifier, to indicate that requests and responses are connected. We then give a list of attributes. We do not give values for most attributes, as we wish to concentrate on the fragmentation behavior rather than packet contents. Attribute values are given for attributes relevant to the fragmentation process. Where "long extended" attributes are used, we indicate the M (More) and T (Truncation) flags as optional square brackets after the attribute name. As no "long extended" attributes have yet been defined, we use example attributes, named as "Example-Long-1", etc. For the sake of simplicity, the maximum chunk size is established in terms of the number of attributes (11).

5.1. Pre-Authorization

When the RADIUS Client needs to send a large amount of data to the RADIUS Server, the data to be sent is split into chunks and sent to the RADIUS Server via multiple Access-Request / Access-Accept exchanges. The example below shows this exchange.

The following is an Access-Request that the RADIUS Client intends to send to a RADIUS Server. However, due to a combination of issues (PMTU, large attributes, etc.), the content does not fit into one Access-Request packet.

```
Access-Request
    User-Name
    NAS-Identifier
    Calling-Station-Id
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1
    Example-Long-2 [M]
    Example-Long-2 [M]
    Example-Long-2
```

Figure 1: Desired Access-Request

The RADIUS Client therefore must send the attributes listed above in a series of chunks. The first chunk contains eight (8) attributes from the original Access-Request, and a Frag-Status attribute. Since the last attribute is "Example-Long-1" with the M flag set, the chunking process also sets the T flag in that attribute. The Access-Request is sent with a RADIUS Identifier field having value 23. The Frag-Status attribute has value More-Data-Pending, to indicate that the RADIUS Client wishes to send more data in a subsequent Access-Request. The RADIUS Client also adds a Service-Type attribute, which indicates that it is part of the chunking process. The packet is signed with the Message-Authenticator attribute, completing the maximum number of attributes (11).

```
Access-Request (ID = 23)
    User-Name
    NAS-Identifier
    Calling-Station-Id
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [MT]
    Frag-Status = More-Data-Pending
    Service-Type = Additional-Authorization
    Message-Authenticator
```

Figure 2: Access-Request (Chunk 1)

Compliant RADIUS Servers (i.e., servers implementing fragmentation) receiving this packet will see the Frag-Status attribute and will postpone all authorization and authentication handling until all of the chunks have been received. This postponement also applies to the verification that the Access-Request packet contains some kind of authentication attribute (e.g., User-Password, CHAP-Password, State, or other future attribute), as required by [RFC2865] (see Section 12.2 for more information on this).

Non-compliant RADIUS Servers (i.e., servers not implementing fragmentation) should also see the Service-Type requesting provisioning for an unknown service and return Access-Reject. Other non-compliant RADIUS Servers may return an Access-Reject or Access-Challenge, or they may return an Access-Accept with a particular Service-Type other than Additional-Authorization. Compliant RADIUS Client implementations MUST treat these responses as if they had received Access-Reject instead.

Compliant RADIUS Servers who wish to receive all of the chunks will respond with the following packet. The value of the State here is arbitrary and serves only as a unique token for example purposes. We only note that it MUST be temporally unique to the RADIUS Server.

```
Access-Accept (ID = 23)
    Frag-Status = More-Data-Request
    Service-Type = Additional-Authorization
    State = 0xabc00001
    Message-Authenticator
```

Figure 3: Access-Accept (Chunk 1)

The RADIUS Client will see this response and use the RADIUS Identifier field to associate it with an ongoing chunking session. Compliant RADIUS Clients will then continue the chunking process. Non-compliant RADIUS Clients will never see a response such as this, as they will never send a Frag-Status attribute. The Service-Type attribute is included in the Access-Accept in order to signal that the response is part of the chunking process. This packet therefore does not provision any network service for the end user.

The RADIUS Client continues the process by sending the next chunk, which includes an additional six (6) attributes from the original packet. It again includes the User-Name attribute, so that non-compliant proxies can process the packet (see Section 11.1). It sets the Frag-Status attribute to More-Data-Pending, as more data is pending. It includes a Service-Type, for the reasons described above. It includes the State attribute from the previous Access-Accept. It signs the packet with Message-Authenticator, as there are no authentication attributes in the packet. It uses a new RADIUS Identifier field.

```
Access-Request (ID = 181)
    User-Name
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1
    Example-Long-2 [M]
    Example-Long-2 [MT]
    Frag-Status = More-Data-Pending
    Service-Type = Additional-Authorization
    State = 0xabc000001
    Message-Authenticator
```

Figure 4: Access-Request (Chunk 2)

Compliant RADIUS Servers receiving this packet will see the Frag-Status attribute and look for a State attribute. Since one exists and it matches a State sent in an Access-Accept, this packet is part of a chunking process. The RADIUS Server will associate the attributes with the previous chunk. Since the Frag-Status attribute has value More-Data-Request, the RADIUS Server will respond with an Access-Accept as before. It MUST include a State attribute, with a value different from the previous Access-Accept. This State MUST again be globally and temporally unique.

```
Access-Accept (ID = 181)
    Frag-Status = More-Data-Request
    Service-Type = Additional-Authorization
    State = 0xdef00002
    Message-Authenticator
```

Figure 5: Access-Accept (Chunk 2)

The RADIUS Client will see this response and use the RADIUS Identifier field to associate it with an ongoing chunking session. The RADIUS Client continues the chunking process by sending the next chunk, with the final attribute(s) from the original packet, and again includes the original User-Name attribute. The Frag-Status attribute is not included in the next Access-Request, as no more chunks are available for sending. The RADIUS Client includes the State attribute from the previous Access-Accept. It signs the packet with Message-Authenticator, as there are no authentication attributes in the packet. It again uses a new RADIUS Identifier field.

```
Access-Request (ID = 241)
    User-Name
    Example-Long-2
    State = 0xdef00002
    Message-Authenticator
```

Figure 6: Access-Request (Chunk 3)

On reception of this last chunk, the RADIUS Server matches it with an ongoing session via the State attribute and sees that there is no Frag-Status attribute present. It then processes the received attributes as if they had been sent in one RADIUS packet. See Section 8.4 for further details on this process. It generates the appropriate response, which can be either Access-Accept or Access-Reject. In this example, we show an Access-Accept. The RADIUS Server MUST send a State attribute, which allows linking the received data with the authentication process.

```
Access-Accept (ID = 241)
    State = 0x98700003
    Message-Authenticator
```

Figure 7: Access-Accept (Chunk 3)

The above example shows in practice how the chunking process works. We reiterate the implementation and security requirements here.

Each chunk is a valid RADIUS packet (see Section 12.2 for some considerations about this), and all RADIUS format and security requirements MUST be followed before any chunking process is applied.

Every chunk except for the last one from a RADIUS Client MUST include a Frag-Status attribute, with value More-Data-Pending. The last chunk MUST NOT contain a Frag-Status attribute. Each chunk except for the last one from a RADIUS Client MUST include a Service-Type attribute, with value Additional-Authorization. Each chunk MUST include a User-Name attribute, which MUST be identical in all chunks. Each chunk except for the first one from a RADIUS Client MUST include a State attribute, which MUST be copied from a previous Access-Accept.

Each Access-Accept MUST include a State attribute. The value for this attribute MUST change in every new Access-Accept and MUST be globally and temporally unique.

5.2. Post-Authorization

When the RADIUS Server wants to send a large amount of authorization data to the RADIUS Client after authentication, the operation is very similar to the pre-authorization process. The presence of a Service-Type = Additional-Authorization attribute ensures that a RADIUS Client not supporting this specification will treat that unrecognized Service-Type as though an Access-Reject had been received instead ([RFC2865], Section 5.6). If the original large Access-Accept packet contained a Service-Type attribute, it will be included with its original value in the last transmitted chunk, to avoid confusion with the one used for fragmentation signaling. It is RECOMMENDED that RADIUS Servers include a State attribute in their original Access-Accept packets, even if fragmentation is not taking place, to allow the RADIUS Client to send additional authorization data in subsequent exchanges. This State attribute would be included in the last transmitted chunk, to avoid confusion with the ones used for fragmentation signaling.

Clients supporting this specification MUST include a Frag-Status = Fragmentation-Supported attribute in the first Access-Request sent to the RADIUS Server, in order to indicate that they would accept fragmented data from the server. This is not required if the pre-authorization process was carried out, as it is implicit.

The following is an Access-Accept that the RADIUS Server intends to send to a RADIUS Client. However, due to a combination of issues (PMTU, large attributes, etc.), the content does not fit into one Access-Accept packet.

```
Access-Accept
    User-Name
    EAP-Message
    Service-Type = Login
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1
    Example-Long-2 [M]
    Example-Long-2 [M]
    Example-Long-2
    State = 0xcba00003
```

Figure 8: Desired Access-Accept

The RADIUS Server therefore must send the attributes listed above in a series of chunks. The first chunk contains seven (7) attributes from the original Access-Accept, and a Frag-Status attribute. Since the last attribute is "Example-Long-1" with the M flag set, the chunking process also sets the T flag in that attribute. The Access-Accept is sent with a RADIUS Identifier field having value 30, corresponding to a previous Access-Request not depicted. The Frag-Status attribute has value More-Data-Pending, to indicate that the RADIUS Server wishes to send more data in a subsequent Access-Accept. The RADIUS Server also adds a Service-Type attribute with value Additional-Authorization, which indicates that it is part of the chunking process. Note that the original Service-Type is not included in this chunk. Finally, a State attribute is included to allow matching subsequent requests with this conversation, and the packet is signed with the Message-Authenticator attribute, completing the maximum number of attributes (11).

```
Access-Accept (ID = 30)
    User-Name
    EAP-Message
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [MT]
    Frag-Status = More-Data-Pending
    Service-Type = Additional-Authorization
    State = 0xcba00004
    Message-Authenticator
```

Figure 9: Access-Accept (Chunk 1)

Compliant RADIUS Clients receiving this packet will see the Frag-Status attribute and suspend all authorization handling until all of the chunks have been received. Non-compliant RADIUS Clients should also see the Service-Type indicating the provisioning for an unknown service and will treat it as an Access-Reject.

RADIUS Clients who wish to receive all of the chunks will respond with the following packet, where the value of the State attribute is taken from the received Access-Accept. They will also include the User-Name attribute so that non-compliant proxies can process the packet (Section 11.1).

```
Access-Request (ID = 131)
    User-Name
    Frag-Status = More-Data-Request
    Service-Type = Additional-Authorization
    State = 0xcba00004
    Message-Authenticator
```

Figure 10: Access-Request (Chunk 1)

The RADIUS Server receives this request and uses the State attribute to associate it with an ongoing chunking session. Compliant RADIUS Servers will then continue the chunking process. Non-compliant RADIUS Servers will never see a response such as this, as they will never send a Frag-Status attribute.

The RADIUS Server continues the chunking process by sending the next chunk, with the final attribute(s) from the original packet. The value of the Identifier field is taken from the received Access-Request. A Frag-Status attribute is not included in the next Access-Accept, as no more chunks are available for sending. The

RADIUS Server includes the original State attribute to allow the RADIUS Client to send additional authorization data. The original Service-Type attribute is included as well.

```
Access-Accept (ID = 131)
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1 [M]
    Example-Long-1
    Example-Long-2 [M]
    Example-Long-2 [M]
    Example-Long-2
    Service-Type = Login
    State = 0xfda000003
    Message-Authenticator
```

Figure 11: Access-Accept (Chunk 2)

On reception of this last chunk, the RADIUS Client matches it with an ongoing session via the Identifier field and sees that there is no Frag-Status attribute present. It then processes the received attributes as if they had been sent in one RADIUS packet. See Section 8.4 for further details on this process.

6. Chunk Size

In an ideal scenario, each intermediate chunk would be exactly the size limit in length. 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 in every chunk.

2. RADIUS fragmentation requires the introduction of some extra attributes for signaling. Specifically, a Frag-Status attribute (7 bytes) is included in 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 RADIUS Server to bind an Access-Request with a previous Access-Challenge. User-Name attributes (from 3 to 255 bytes) are included in every chunk the RADIUS Client sends, as they are required by the proxies to route the packet to its destination. Together, these attributes can generate from up to 13 to 517 bytes of signaling data, reducing the amount of payload information that can be sent in each chunk.

3. RADIUS packets SHOULD be adjusted to avoid exceeding the network MTU. Otherwise, IP fragmentation may occur, with 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 in the chunk. This is described in further detail in Section 8.1.

7. Allowed Large Packet Size

There are no provisions for signaling how much data is to be sent via the fragmentation process as a whole. It is difficult to define what is meant by the "length" of any fragmented data. That data can be multiple attributes and can include RADIUS attribute header fields, or it can be one or more "large" attributes (more than 256 bytes in length). Proxies can also filter these attributes, to modify, add, or delete them and their contents. These proxies act on a "packet by packet" basis and cannot know what kind of filtering actions they will take on future packets. As a result, it is impossible to signal any meaningful value for the total amount of additional data.

Unauthenticated end users are permitted to trigger the exchange of large amounts of fragmented data between the RADIUS Client and the RADIUS Server, having the potential to allow denial-of-service (DoS) attacks. An attacker could initiate a large number of connections, each of which requests the RADIUS Server to store a large amount of data. This data could cause memory exhaustion on the RADIUS Server and result in authentic users being denied access. It is worth noting that authentication mechanisms are already designed to avoid exceeding the size limit.

Hence, implementations of this specification MUST limit the total amount of data they send and/or receive via this specification. Its default value SHOULD be 100 kilobytes. Any more than this may turn RADIUS into a generic transport protocol, which is undesirable. This limit SHOULD be configurable, so that it can be changed if necessary.

Implementations of this specification MUST limit the total number of round trips used during the fragmentation process. Its default value SHOULD be 25. Any more than this may indicate an implementation error, misconfiguration, or DoS attack. This limit SHOULD be configurable, so that it can be changed if necessary.

For instance, let's imagine that the RADIUS Server wants to transport a SAML assertion that is 15000 bytes long to the RADIUS Client. In this hypothetical scenario, we assume that there are three intermediate proxies, each one inserting a Proxy-State attribute of 20 bytes. Also, we assume that the State attributes generated by the RADIUS Server have a size of 6 bytes and the User-Name attribute takes 50 bytes. Therefore, the amount of free space in a chunk for the transport of the SAML assertion attributes is as follows: Total (4096 bytes) - RADIUS header (20 bytes) - User-Name (50 bytes) - Frag-Status (7 bytes) - Service-Type (6 bytes) - State (6 bytes) - Proxy-State (20 bytes) - Proxy-State (20 bytes) - Proxy-State (20 bytes) - Message-Authenticator (18 bytes), resulting in a total of 3929 bytes. This amount of free space allows the transmission of up to 15 attributes of 255 bytes each.

According to [RFC6929], a Long-Extended-Type provides a payload of 251 bytes. Therefore, the SAML assertion described above would result in 60 attributes, requiring four round trips to be completely transmitted.

8. Handling Special Attributes

8.1. Proxy-State Attribute

RADIUS proxies may introduce Proxy-State attributes into any Access-Request packet they forward. If they are unable to add this information to the packet, they may silently discard it rather than forward it to its destination; this would lead to DoS situations. Moreover, any Proxy-State attribute received by a RADIUS Server in an Access-Request packet MUST be copied into the corresponding reply packet. For these reasons, Proxy-State attributes require 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 in the reply packet. This means that the RADIUS 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 much space MUST be left in 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, RADIUS Clients need a mechanism to discover the amount of

space required by proxies to introduce their Proxy-State attributes. In the following paragraphs, we describe a new mechanism to perform such a discovery:

1. When a RADIUS Client does not know how much space will be required by intermediate proxies for including their Proxy-State attributes, it SHOULD start using a conservative value (e.g., 1024 bytes) as the chunk size.

2. When the RADIUS Server receives a chunk from the RADIUS 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 RADIUS Client in the next reply packet, encoded into a new attribute called Proxy-State-Length. The RADIUS Server MAY artificially increase this quantity in order to handle situations where proxies behave inconsistently (e.g., they generate Proxy-State attributes with a different size for each packet) or where intermediary proxies remove Proxy-State attributes generated by other proxies. Increasing this value would make the RADIUS Client leave some free space for these situations.

3. The RADIUS Client SHOULD respond to the reception of this attribute by adjusting the maximum size for the next chunk accordingly. However, as the Proxy-State-Length offers just an estimation of the space required by the proxies, the RADIUS Client MAY select a smaller amount in environments known to be problematic.

8.2. 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 to the fragmentation. If the procedure described in Section 5 is followed, two different State attributes could be included in a single chunk. This is something explicitly forbidden in [RFC2865].

A straightforward solution consists of making the RADIUS Server send the original State attribute in the last chunk of the sequence (attributes can be reordered 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 in it. The RADIUS Client MUST ensure that this original State is included in the first chunk sent to the RADIUS Server (as this one never contains any State attribute due to fragmentation).

8.3. Service-Type Attribute

This RADIUS fragmentation mechanism makes use of the Service-Type attribute to indicate that an Access-Accept packet is not granting access to the service yet, since an additional authorization exchange needs to be performed. Similarly to the State attribute, the RADIUS Server has to send the original Service-Type attribute in the last Access-Accept of the RADIUS conversation to avoid ambiguity.

8.4. Rebuilding the Original Large Packet

The RADIUS Client stores the RADIUS attributes received in each chunk in a list, 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 that list, as they have been introduced as part of the fragmentation signaling and hence are not part of the original packet.

- State (except the one in the last chunk, if present)
- Service-Type = Additional-Authorization
- Frag-Status
- Proxy-State-Length

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)
- Service-Type = Additional-Authorization
- Frag-Status
- Proxy-State (except the ones in the last chunk)
- User-Name (except the one in the first chunk)

9. New T Flag for the Long Extended Type Attribute Definition

This document defines a new field in the Long Extended Type attribute format. This field is one bit in size and is called "T" for Truncation. It indicates that the attribute is intentionally truncated in this chunk and is to be continued in the next chunk of the sequence. The combination of the M flag and the T flag indicates that the attribute is fragmented (M flag) but that all the fragments are not available in this chunk (T flag). Proxies implementing [RFC6929] will see these attributes as invalid (they will not be able to reconstruct them), but they will still forward them, as Section 5.2 of [RFC6929] indicates that they SHOULD forward unknown attributes anyway.

As a consequence of this addition, the Reserved field is now 6 bits long (see Section 12.1 for some considerations). The following figure represents the new attribute format:

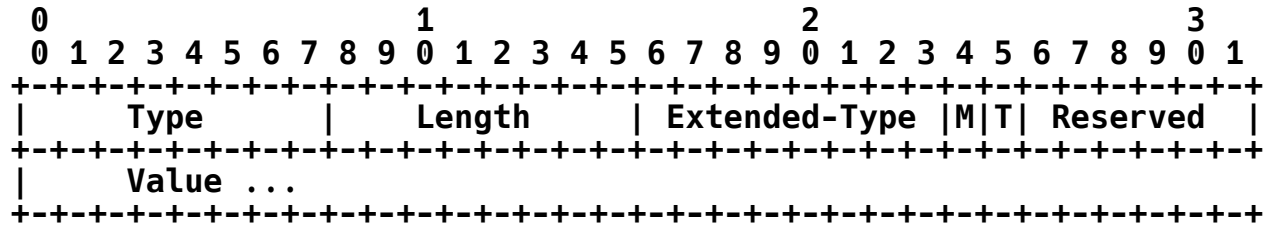


Figure 12: Updated Long Extended Type Attribute Format

10. New Attribute Definition

This document proposes the definition of two new extended type attributes, called Frag-Status and Proxy-State-Length. The format of these attributes follows the indications for an Extended Type attribute defined in [RFC6929].

10.1. Frag-Status Attribute

This attribute is used for fragmentation signaling, and its meaning depends on the code value transported within it. The following figure represents the format of the Frag-Status attribute:

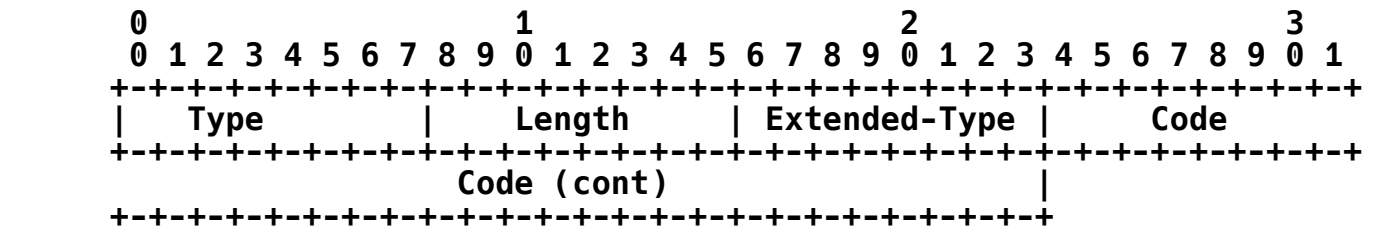


Figure 13: Frag-Status Format

Type

241

Length

7

Extended-Type

1

Code

4 bytes. Integer indicating the code. The values defined in this specification are:

- 0 - Reserved
- 1 - Fragmentation-Supported
- 2 - More-Data-Pending
- 3 - More-Data-Request

This attribute MAY be present in Access-Request, Access-Challenge, and Access-Accept packets. It MUST NOT be included in Access-Reject packets. RADIUS Clients supporting this specification MUST include a Frag-Status = Fragmentation-Supported attribute in the first Access-Request sent to the RADIUS Server, in order to indicate that they would accept fragmented data from the server.

10.2. Proxy-State-Length Attribute

This attribute indicates to the RADIUS Client the length of the Proxy-State attributes received by the RADIUS Server. This information is useful for adjusting the length of the chunks sent by the RADIUS Client. The format of this Proxy-State-Length attribute is as follows:

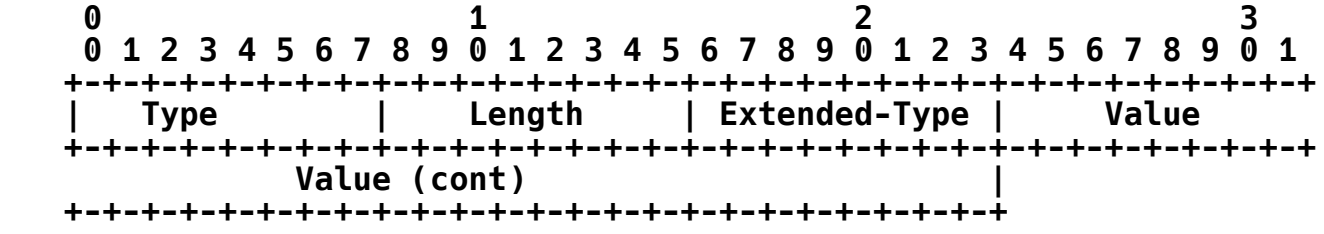


Figure 14: Proxy-State-Length Format

Type

241

Length

7

Extended-Type

2

Value

4 bytes. Total length (in bytes) of received Proxy-State attributes (including headers). As the RADIUS Length field cannot take values over 4096 bytes, values of Proxy-State-Length MUST be less than that maximum length.

This attribute MAY be present in Access-Challenge and Access-Accept packets. It MUST NOT be included in Access-Request or Access-Reject packets.

10.3. Table of Attributes

The following table shows the different attributes defined in this document, along with the types of RADIUS packets in which they can be present.

	Type of Packet			
Attribute Name	Req	Acc	Rej	Cha
Frag-Status	0-1	0-1	0	0-1
Proxy-State-Length	0	0-1	0	0-1

11. Operation with Proxies

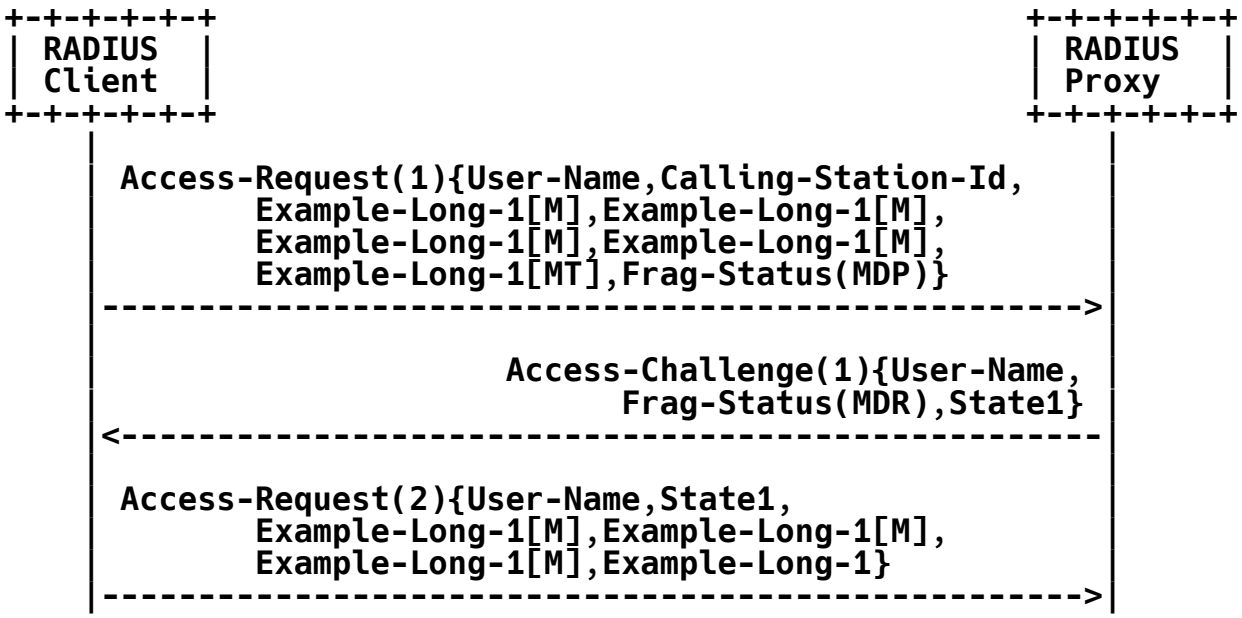
The fragmentation mechanism defined above is designed to be transparent to legacy proxies, as long as they do not want to modify any fragmented attribute. Nevertheless, updated proxies supporting this specification can even modify fragmented attributes.

11.1. Legacy Proxies

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

11.2. Updated Proxies

Updated proxies can interact with RADIUS Clients and Servers in order to obtain the complete large packet before starting to forward 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 example in Figure 15 shows how an updated proxy interacts with the RADIUS Client to (1) obtain a large Access-Request packet and (2) modify an attribute, resulting in an even larger packet. The proxy then interacts with the RADIUS Server to complete the transmission of the modified packet, as shown in Figure 16.



Proxy Modifies Attribute Data, Increasing Its Size from 9 Fragments to 11 Fragments

Figure 15: Updated Proxy Interacts with RADIUS Client

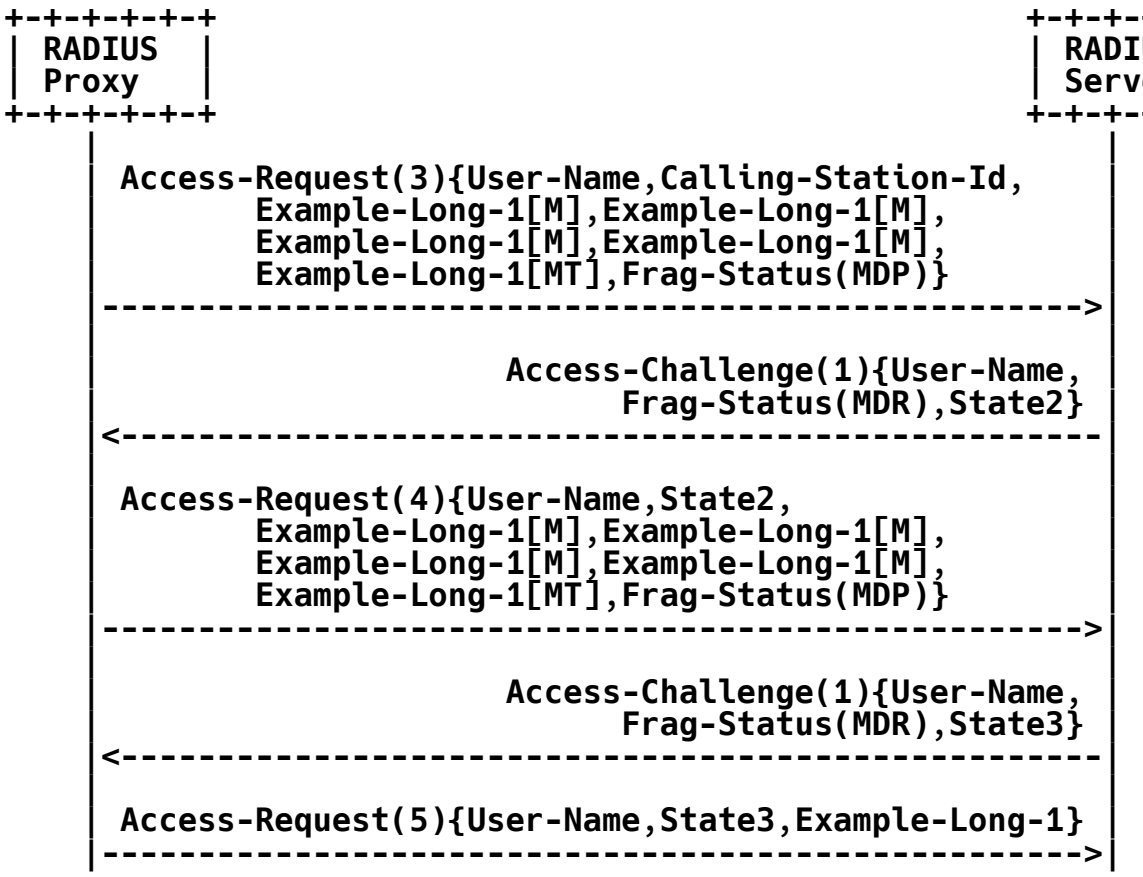


Figure 16: Updated Proxy Interacts with RADIUS Server

12. General Considerations

12.1. T Flag

As described in Section 9, this document modifies the definition of the Reserved field of the Long Extended Type attribute [RFC6929] by allocating an additional flag called the T flag. The meaning and position of this flag are defined in this document, and nowhere else. This might cause an issue if subsequent specifications want to allocate a new flag as well, as there would be no direct way for them to know which parts of the Reserved field have already been defined.

An immediate and reasonable solution for this issue would be declaring that this RFC updates [RFC6929]. In this way, [RFC6929] would include an "Updated by" clause that will point readers to this document. Another alternative would be creating an IANA registry for

the Reserved field. However, the RADIUS Extensions (RADEXT) working group thinks that would be overkill, as a large number of specifications extending that field are not expected.

In the end, the proposed solution is that this experimental RFC should not update RFC 6929. Instead, we rely on the collective mind of the working group to remember that this T flag is being used as specified by this Experimental document. If the experiment is successful, the T flag will be properly assigned.

12.2. Violation of RFC 2865

Section 5.1 indicates that all authorization and authentication handling will be postponed until all the chunks have been received. This postponement also applies to the verification that the Access-Request packet contains some kind of authentication attribute (e.g., User-Password, CHAP-Password, State, or other future attribute), as required by [RFC2865]. This checking will therefore be delayed until the original large packet has been rebuilt, as some of the chunks may not contain any of them.

The authors acknowledge that this specification violates the "MUST" requirement of [RFC2865], Section 4.1 that states that "An Access-Request MUST contain either a User-Password or a CHAP-Password or a State." We note that a proxy that enforces that requirement would be unable to support future RADIUS authentication extensions. Extensions to the protocol would therefore be impossible to deploy. All known implementations have chosen the philosophy of "be liberal in what you accept." That is, they accept traffic that violates the requirement of [RFC2865], Section 4.1. We therefore expect to see no operational issues with this specification. After we gain more operational experience with this specification, it can be reissued as a Standards Track document and can update [RFC2865].

12.3. Proxying Based on User-Name

This proposal assumes that legacy proxies base their routing decisions on the value of the User-Name attribute. For this reason, every packet sent from the RADIUS Client to the RADIUS Server (either chunks or requests for more chunks) MUST contain a User-Name attribute.

12.4. Transport Behavior

This proposal does not modify the way RADIUS interacts with the underlying transport (UDP). That is, RADIUS keeps following a lock-step behavior that requires receiving an explicit acknowledgement for each chunk sent. Hence, bursts of traffic that could congest links between peers are not an issue.

Another benefit of the lock-step nature of RADIUS is that there are no security issues with overlapping fragments. Each chunk simply has a length, with no Fragment Offset field as with IPv4. The order of the fragments is determined by the order in which they are received. There is no ambiguity about the size or placement of each chunk, and therefore no security issues associated with overlapping chunks.

13. Security Considerations

As noted in many earlier specifications ([RFC5080], [RFC6158], etc.), RADIUS security is problematic. This specification changes nothing related to the security of the RADIUS protocol. It requires that all Access-Request packets associated with fragmentation are authenticated using the existing Message-Authenticator attribute. This signature prevents forging and replay, to the limits of the existing security.

The ability to send bulk data from one party to another creates new security considerations. RADIUS Clients and Servers may have to store large amounts of data per session. The amount of this data can be significant, leading to the potential for resource exhaustion. We therefore suggest that implementations limit the amount of bulk data stored per session. The exact method for this limitation is implementation-specific. Section 7 gives some indications of what could be reasonable limits.

The bulk data can often be pushed off to storage methods other than the memory of the RADIUS implementation. For example, it can be stored in an external database or in files. This approach mitigates the resource exhaustion issue, as RADIUS Servers today already store large amounts of accounting data.

14. IANA Considerations

The Internet Assigned Numbers Authority (IANA) has registered the Attribute Types and Attribute Values defined in this document in the RADIUS namespaces as described in the "IANA Considerations" section of [RFC3575], in accordance with BCP 26 [RFC5226]. For RADIUS packets, attributes, and registries created by this document, IANA has updated <http://www.iana.org/assignments/radius-types> accordingly.

In particular, this document defines two new RADIUS attributes, entitled "Frag-Status" (value 241.1) and "Proxy-State-Length" (value 241.2), which have been allocated from the short extended space as described in [RFC6929]:

Type	Name	Length	Meaning
241.1	Frag-Status	7	Signals fragmentation
241.2	Proxy-State-Length	7	Indicates the length of the received Proxy-State attributes

The Frag-Status attribute also defines an 8-bit "Code" field, for which IANA has created and now maintains a new sub-registry entitled "Code Values for RADIUS Attribute 241.1, Frag-Status". Initial values for the RADIUS Frag-Status "Code" registry are given below; future assignments are to be made through "RFC Required" [RFC5226]. Assignments consist of a Frag-Status "Code" name and its associated value.

Value	Frag-Status Code Name	Definition
0	Reserved	See Section 10.1
1	Fragmentation-Supported	See Section 10.1
2	More-Data-Pending	See Section 10.1
3	More-Data-Request	See Section 10.1
4-255	Unassigned	

Additionally, IANA has allocated a new Service-Type value for "Additional-Authorization".

Value	Service Type Value	Definition
19	Additional-Authorization	See Section 5.1

15. References

15.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000, <http://www.rfc-editor.org/info/rfc2865>.

[RFC3575] Aboba, B., "IANA Considerations for RADIUS (Remote Authentication Dial In User Service)", RFC 3575, July 2003, <http://www.rfc-editor.org/info/rfc3575>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC6158] DeKok, A., Ed., and G. Weber, "RADIUS Design Guidelines", BCP 158, RFC 6158, March 2011, <http://www.rfc-editor.org/info/rfc6158>.

[RFC6929] DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", RFC 6929, April 2013, <http://www.rfc-editor.org/info/rfc6929>.

15.2. Informative References

[ABFAB-Arch] Howlett, J., Hartman, S., Tschofenig, H., Lear, E., and J. Schaad, "Application Bridging for Federated Access Beyond Web (ABFAB) Architecture", Work in Progress, draft-ietf-abfab-arch-13, July 2014.

[RADIUS-Larger-Pkts] Hartman, S., "Larger Packets for RADIUS over TCP", Work in Progress, draft-ietf-radext-bigger-packets-03, March 2015.

[RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, June 2000, <http://www.rfc-editor.org/info/rfc2866>.

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003, <http://www.rfc-editor.org/info/rfc3579>.

[RFC4849] Congdon, P., Sanchez, M., and B. Aboba, "RADIUS Filter Rule Attribute", RFC 4849, April 2007, <http://www.rfc-editor.org/info/rfc4849>.

[RFC5080] Nelson, D. and A. DeKok, "Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes", RFC 5080, December 2007, <http://www.rfc-editor.org/info/rfc5080>.

[RFC5176] Chiba, M., Dommety, G., Eklund, M., Mitton, D., and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January 2008, <http://www.rfc-editor.org/info/rfc5176>.

[SAML-RADIUS] Howlett, J., Hartman, S., and A. Perez-Mendez, Ed., "A RADIUS Attribute, Binding, Profiles, Name Identifier Format, and Confirmation Methods for SAML", Work in Progress, draft-ietf-abfab-aaa-saml-10, February 2015.

Acknowledgements

The authors would like to thank the members of the RADEXT working group who have contributed to the development of this specification by either participating in the discussions on the mailing lists or sending comments about our RFC.

The authors also thank David Cuenca (University of Murcia) for implementing a proof-of-concept implementation of this RFC that has been useful to improve the quality of the specification.

This work has been partly funded by the GEANT GN3+ SA5 and CLASSe (<http://www.um.es/classe/>) projects.

Authors' Addresses

Alejandro Perez-Mendez (editor)
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 SARL
57bis Boulevard des Alpes
Meylan 38240
France

EMail: aland@networkradius.com
URI: http://networkradius.com