RADIUS EXTensions Working Group
Internet-Draft
Updates: RFC6929 (if approved)
Intended status: Experimental
Expires: October 9, 2014

A. Perez-Mendez
R. Marin-Lopez
F. Pereniguez-Garcia
G. Lopez-Millan
University of Murcia
D. Lopez
Telefonica I+D
A. DeKok
Network RADIUS
April 7, 2014

Support of fragmentation of RADIUS packets
draft-ietf-radext-radius-fragmentation-06

Abstract

The Remote Authentication Dial-In User Service (RADIUS) protocol is limited to a total packet size of 4096 octets. Provisions exist for fragmenting large amounts of authentication data across multiple packets, via Access-Challenge. 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 are designed to be invisible to proxies, and "fail-safe" to legacy clients and servers.

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 October 9, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
1.1. Requirements Language 4
2. Scope of this document 4
3. Overview 6
4. Fragmentation of packets 8
4.1. Pre-authorization 9
4.2. Post-authorization 13
5. Chunk size 16
6. Allowed large packet size 17
7. Handling special attributes 18
7.1. Proxy-State attribute 18
7.2. State attribute 19
7.3. Service-Type attribute 20
7.4. Rebuilding the original large packet 20
8. New flag T field for the Long Extended Type attribute definition 20
9. New attribute definition 21
9.1. Frag-Status attribute 21
9.2. Proxy-State-Len attribute 22
9.3. Table of attributes 23
10. Operation with proxies 23
10.1. Legacy proxies 23
10.2. Updated proxies 24
11. Operational considerations 25
11.1. Flag T 25
11.2. Violation of RFC2865 26
11.3. Proxying based on User-Name 26
12. Security Considerations 26
13. IANA Considerations 27
14. Acknowledgements 27
15. References 28
15.1. Normative References 28
15.2. Informative References 28
Authors' Addresses 28

1. Introduction

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

The above packet size limitation mean that peers desiring to send large amounts of data must fragment it across multiple packets. For example, RADIUS-EAP [RFC3579] defines how an 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 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 server is large and highly dynamic. In other cases, the NAS needs to send large amounts of authorization data to the server. Both of these cases are un-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 octets. The combination of the two limitations means that there is a pressing need for a method to send large amounts of authorization data between NAS and AS, with no accompanying solution.

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

- The first approach does not talk about large amounts of data sent from the NAS to a server. Leveraging EAP (request/challenge) to send the data is not feasible, as EAP already fills packet to PMTU, and not all authentications use EAP. Moreover, as noted for NAS-Filter-Rule ([RFC4849]), this approach does entirely solve the problem of sending large amounts of data from a server to a NAS.

- The second approach 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. SAML sentences 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 set up of an additional trust infrastructure (e.g. PKI) would be required to allow secure retrieving of the information from the web server.

- o PMTU discovery does not solve the problem, as it does not allow to send data larger than the minimum of (PMTU or 4096) octets.

This document provides a mechanism to allow RADIUS peers to exchange large amounts of authorization data exceeding the 4096 octet limit, by fragmenting it across several client/server 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 in [RFC6929]). It is also transparent to existing RADIUS proxies, which do not implement this specification. The only systems needing to implement the draft are the ones which 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 re-assemble the data in order to either examine and/or modify it.

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. Scope of this document

This specification describes how a RADIUS client and a RADIUS server can exchange data exceeding the 4096 octet limit imposed by one packet. However, the mechanism described in this specification MUST NOT be used to exchange more than 100K of data. It has not been designed to substitute for stream-oriented transport protocols, such as TCP or SCTP. Experience shows that attempts to transport bulk data across the Internet with UDP will inevitably fail, unless they re-implement all of the behavior of TCP. The underlying design of RADIUS lacks the proper retransmission policies or congestion control mechanisms which would make it a competitor to TCP.

Therefore, RADIUS/UDP transport is by design unable to transport bulk data. It is both undesired and impossible to change the protocol at this point in time. This specification is intended to allow the

transport of slightly more than 4096 octets 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 6, below, describes with further details the reasoning for this limitation, and recommends administrators to adjust it 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 of 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, its preferable to not introduce any modification to their operation 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 4K 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 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 CoA packets. Instead, the CoA client MUST 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, which is 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, with the corresponding possibility for implementation issues, mistakes, etc.

Where CoA clients need to send large amounts of authorization data to a NAS, they need only send a minimal CoA-Request packet, containing Service-Type of Authorize-Only, as per RFC 5176. They SHOULD also have a co-located RADIUS server, for the sole purpose of implementing this specification.

The NAS will then perform fragmentation as per this draft to the RADIUS server it is configured to use. That RADIUS server SHOULD then act as a proxy, and forward the Access-Request to the RADIUS server on the CoA client. That RADIUS server can then send the large amounts of authorization to the proxy, which then sends them to the NAS.

That is, the NAS sends packets to a server which proxies them to the system which is co-located with the CoA client.This process 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 is preferred to unspecified communication methods and multiple ways of achieving the same result.

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.

When a sessions authorization is changed, the CoA server MUST continue the existing service until the new authorization parameters are applied. The change of service SHOULD be done atomically. If the CoA server is unable to apply the new authorization, it MUST terminate the user session.

3. 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 the "pre-authorization" as the exchange of authorization information before the end user authentication has started, while the term "post-authorization" is used to refer to an authorization exchange happening after this authentication process.

In this specification we refer to the "size limit" as the practical limit on RADIUS packet sizes. This limit is the minimum of 4096 octets, and the current PMTU. We define below a method which uses Access-Request and Access-Accept in order to exchange fragmented data. The NAS 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 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 10.1). 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 NAS in a later Access-Request. Each Access-Accept contains a Service-Type indicating that the service being provided is fragmentation, and that the Access-Accept should not be interpreted as providing network access to the end user.

When a RADIUS client or server need 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 client and server, and the overhead for fragmentation signalling attributes. Specific details are given in Section 5. A a new attribute called Frag-Status (Section 9.1) 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.

When multiple chunks are sent, a special situation may occur for 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 flag "T" in the Reserved field of the "Long Extended Type" attribute format (see Section 8 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. There are neither integrity nor security checks 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 which simply forwards packets without detailed examination or any modification will be able to proxy "chunks".

4. Fragmentation of packets

When the NAS or the AS 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 following sections describe how to perform fragmentation for packets from the NAS to the server, followed by packets from the server to the NAS. 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 behaviour, 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. The maximum chunk size is established in term of number of attributes (11), for sake of simplicity.

4.1. Pre-authorization

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

The following is an Access-Request which the NAS intends to send to a 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 NAS 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 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 NAS wishes to send more data in a subsequent Access-Request. The NAS 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 servers (i.e. servers implementing fragmentation) receiving this packet will see the Frag-Status attribute, and postpone all authorization and authentication handling until all of the chunks have been received. This postponement also affects 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 11.2 for more information on this).

Non-compliant 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 servers may return an Access-Reject, Access-Challenge, or an Access-Accept with a particular Service-Type other then Additional-Authorization. Compliant NAS implementations MUST treat these responses as if they had received Access-Reject instead.

Compliant 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 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 NAS will see this response, and use the RADIUS Identifier field to associate it with an ongoing chunking session. Compliant NASes will then continue the chunking process. Non-compliant NASes 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 NAS 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 10.1). It sets the Frag-Status attribute to More-Data-Pending, as more data is pending. It includes a Service-Type for 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 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 server will associate the attributes with the previous chunk. Since the Frag-Status attribute has value More-Data-Request, the 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 NAS will see this response, and use the RADIUS Identifier field to associate it with an ongoing chunking session. The NAS 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 NAS 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 server matches it with an ongoing session via the State attribute, and sees that there is no Frag-Status attribute present. It then process the received attributes as if they had been sent in one RADIUS packet. See Section 7.4 for further details of 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 server MUST send a State attribute, which permits link 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 re-iterate the implementation and security requirements here.

Each chunk is a valid RADIUS packet, 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 NAS 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 from a NAS 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 NAS 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.

4.2. Post-authorization

When the AS wants to send a large amount of authorization data to the NAS after authentication, the operation is very similar to the pre-authorization one. The presence of Service-Type = Additional-Authorization attribute ensures that a NAS 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 signalling. It is strongly RECOMMENDED that servers include a State attribute on their original Access-Accept packets, even if fragmentation is not taking place, to allow the 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 signalling.

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

The following is an Access-Accept which the AS intends to send to a 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 [M]
    Example-Long-1
    Example-Long-2 [M]
    Example-Long-2 [M]
    Example-Long-2
    State = 0xcba00003
```

Figure 8: Desired Access-Accept

The AS 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 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 AS wishes to send more data in a subsequent Access-Accept. The AS 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 of 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 clients receiving this packet will see the Frag-Status attribute, wand suspend all authorization and authentication handling until all of the chunks have been received. Non-compliant clients should also see the Service-Type indicating the provisioning for an unknown service, and will treat it as an Access-Reject.

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 also include the User-Name attribute so that non-compliant proxies can process the packet (Section 10.1).

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

The AS 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 AS includes the original State attribute to allow the 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 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 7.4 for further details of this process.

5. 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 octets of spare space on every chunk.

2. RADIUS fragmentation requires the introduction of some extra attributes for signalling. Specifically, a Frag-Status attribute (7 octets) is included on every chunk of a packet, except the last one. A RADIUS State attribute (from 3 to 255 octets) 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 octets) 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 13 to 517 octets of signalling 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 7.1.

6. Allowed large packet size

There are no provisions for signalling 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, which includes RADIUS attribute header fields. Or it can be one or more "large" attributes (more than 256 octets 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 take on future packets. As a result, it is impossible to signal any meaningful value for the total amount of additional data.

Unauthenticated clients are permitted to trigger the exchange of large amounts of fragmented data between the NAS and the AS, having the potential to allow Denial of Service (DoS) attacks. An attacker could initiate a large number of connections, each of which requests the server to store a large amount of data. This data could cause memory exhaustion on the 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 to 100K. Any more than this may turn RADIUS into a generic transport protocol, which is undesired. It is RECOMMENDED that this limit be exposed to administrators, 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 to 25. Any more than this may indicate an implementation error, misconfiguration, or a denial of service (DoS) attack. It is RECOMMENDED that this limit be exposed to administrators, so that it can be changed if necessary.

For instance, let's imagine the RADIUS server wants to transport an SAML assertion which is 15000 octets long, to the RADIUS client. In this hypothetical scenario, we assume there are 3 intermediate

proxies, each one inserting a Proxy-State attribute of 20 octets. Also we assume the State attributes generated by the RADIUS server have a size of 6 octets. Therefore, the amount of free space in a chunk for the transport of the SAML assertion attributes is: Total (4096) - RADIUS header (20) - Frag-Status (7 octets) - Service-Type (6 octets) - State (6 octets) - Proxy-State (20 octets) - Proxy-State (20) - Proxy-State (20) - Message-Authenticator (18 octets), resulting in a total of 3979 octets, that is, 15 attributes of 255 bytes.

According to [RFC6929], a Long-Extended-Type provides a payload of 251 octets. Therefore, the SAML assertion described above would result into 60 attributes, requiring of 4 round-trips to be completely transmitted.

7. Handling special attributes

7.1. 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 much 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 much space will be required by intermediate proxies for including their Proxy-State

attributes, it SHOULD start using a conservative value (e.g. 1024 octets) 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. The server MAY artificially increase this quantity in order to handle with situations where proxies behave inconsistently (e.g. they generate Proxy-State attributes with a different size for each packet), or for situations where intermediary proxies remove Proxy-State attributes generated by other proxies. Increasing this value would make the client to leave some free space for these situations.

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

7.2. 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 4 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 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).

7.3. Service-Type attribute

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

7.4. 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 signalling and hence, they 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-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)
- Service-Type = Additional-Authorization
- Frag-Status
- Proxy-State (except the ones in the last chunk)
- User-Name (except the one in the first chunk)

8. New flag T field 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 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). Proxies implementing [RFC6929] will see these attributes as invalid (they will not be able to reconstruct them), but they will still forward them as [RFC6929] section 5.2 indicates they SHOULD forward unknown attributes anyway.

As a consequence of this addition, the Reserved field is now 6 bits long (see Section 11.1 for some considerations). The following figure represents the new attribute format.

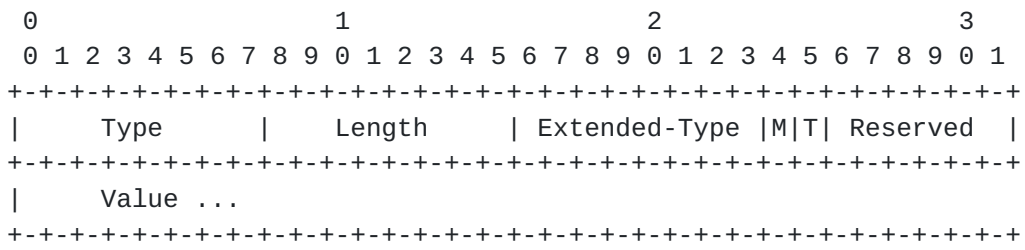


Figure 12: Updated Long Extended Type attribute format

9. New attribute definition

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

9.1. Frag-Status attribute

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

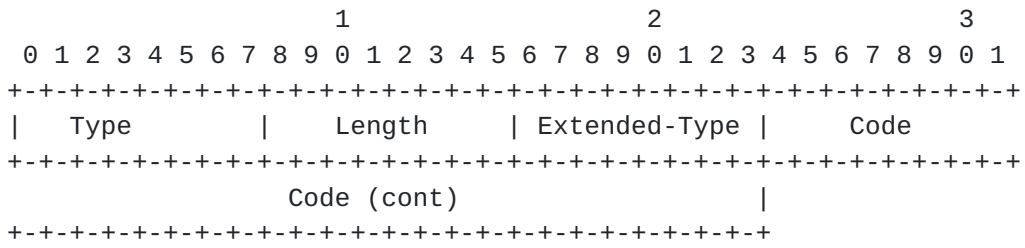


Figure 13: Frag-Status format

Type

To be assigned (TBA)

Length

7

Extended-Type

To be assigned (TBA).

Code

4 byte. Integer indicating the code. The values defined in this specifications 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. Clients supporting this specification MUST include a Frag-Status = Fragmentation-Supported attribute in the first Access-Request sent to the server, in order to indicate they would accept fragmented data from the sever.

9.2. 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:

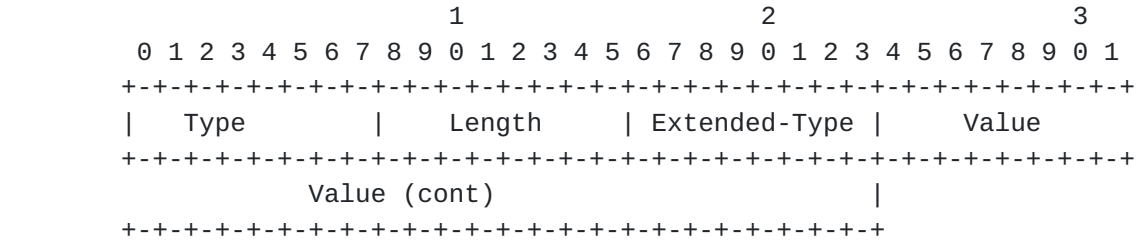


Figure 14: Proxy-State-Len format

Type

To be assigned (TBA)

Length

7

Extended-Type

To be assigned (TBA).

Value

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

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

9.3. 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
Frag-Status	0-1	0-1	0	0-1
Proxy-State-Len	0	0-1	0	0-1

Figure 15

10. 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.

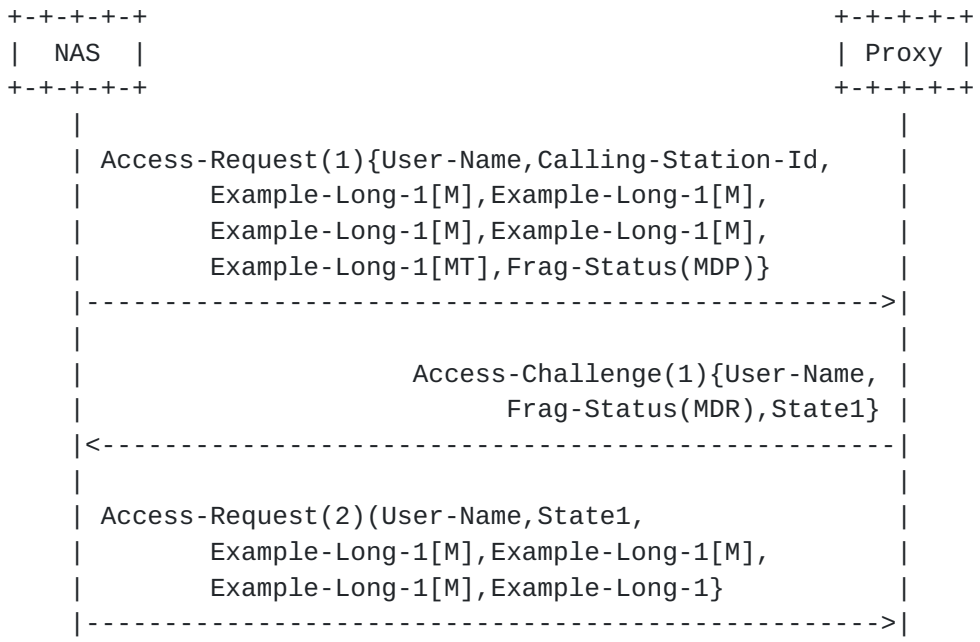
10.1. Legacy proxies

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.

10.2. Updated proxies

Updated proxies can interact with clients and servers in order to obtain the complete large packet before starting 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 16: Updated proxy interacts with NAS

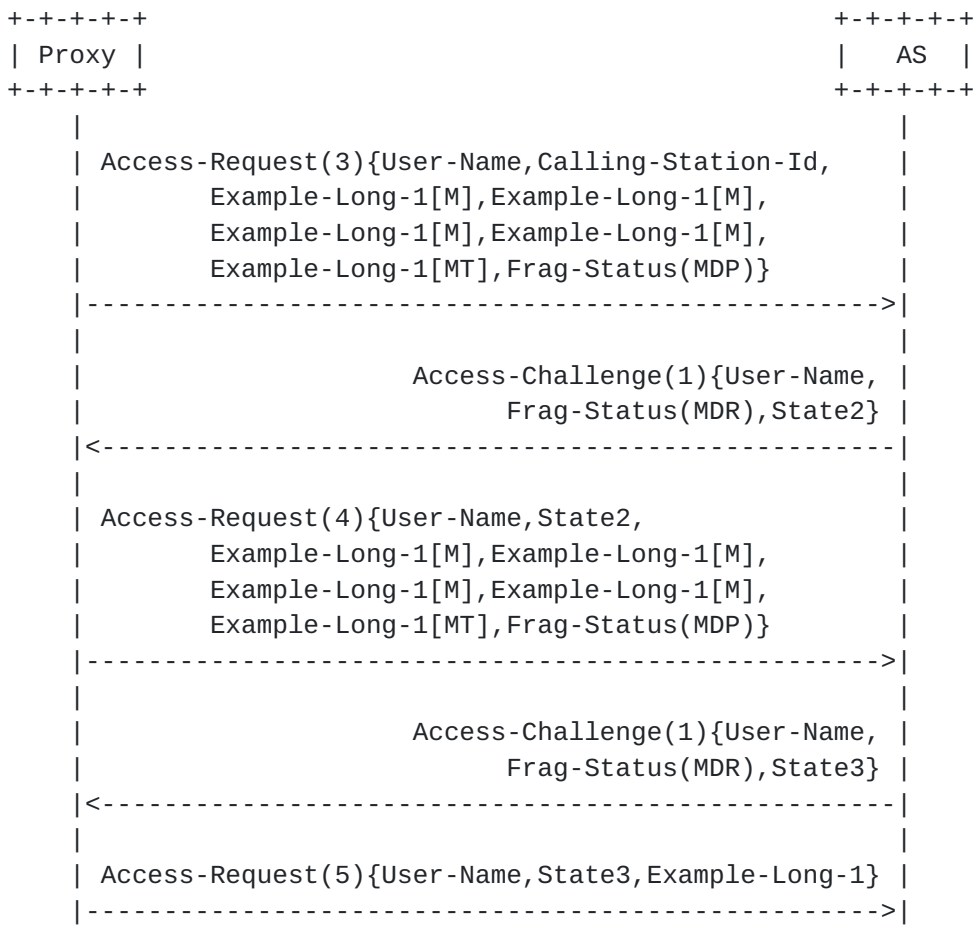


Figure 17: Updated proxy interacts with AS

11. Operational considerations

11.1. Flag T

As described in Section 8, this document modifies the definition of the "Reserved" field of the "Long Extended Type" attribute [RFC6929], by allocating an additional flag "T". The meaning and position of this flag is defined in this document, and nowhere else. This might generate 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 draft updates [RFC6929]. In this way, [RFC6929] would include an "Updated by" clause that will point readers to this document. However, since this draft belongs to the Experimental track and [RFC6929] belongs to the Standards track, we do not know if including that "Updates" clause would be acceptable.

Another alternative would be creating an IANA registry for the "Reserved" field. However, the working group thinks that would be overkill, as not such a great number of specifications extending that field are expected.

Hence, we have decided to include the "Updates" clause in the document so far.

11.2. Violation of RFC2865

Section 4.1 indicates that all authorization and authentication handling will be postponed until all the chunks have been received. This postponement also affects 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 this is formally violating [RFC2865], but there are no known operational issues with it. Proxies are supposed to not check this, as [RFC2865] specifies that other attributes might be considered as authentication information in future extensions, and doing so would make them too restrictive. Once this document goes beyond being considered as experimental, it will state it updates [RFC2865].

11.3. Proxying based on User-Name

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

12. 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. 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 6 gives some indications on 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 servers today already store large amounts of accounting data.

13. IANA Considerations

The authors request that Attribute Types and Attribute Values defined in this document be registered by the Internet Assigned Numbers Authority (IANA) from 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 is requested to place them at http://www.iana.org/assignments/radius-types.

This document defines the following RADIUS messages:

- Frag-Status
- Proxy-State-Len

Additionally, allocation of a new Service-Type value for "Additional-Authorization" is requested.

14. Acknowledgements

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

The authors also thank David Cuenca (University of Murcia) for implementing a proof of concept implementation of this draft 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://sec.cs.kent.ac.uk/CLASSe/) projects.

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.

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

[RFC3575] Aboba, B., "IANA Considerations for RADIUS (Remote Authentication Dial In User Service)", RFC 3575, July 2003.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

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

[RFC6929] DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", RFC 6929, April 2013.

15.2. Informative References

[RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, 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.

[RFC4849] Congdon, P., Sanchez, M., and B. Aboba, "RADIUS Filter Rule Attribute", RFC 4849, April 2007.

[RFC5080] Nelson, D. and A. DeKok, "Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes", RFC 5080, December 2007.

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