

Network Working Group
Internet Draft
Intended Status: Standards Track
Expiration Date: April 21, 2018

E. Chen
N. Shen
Cisco Systems
October 20, 2017

RADIUS Extended Identifier Attribute
draft-chen-radext-identifier-attr-02.txt

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on October 26, 2017.

Copyright Notice

Copyright (c) 2017 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.

Abstract

The limitation with the one-octet "Identifier" field in the RADIUS packet is well known. In this document we propose extensions to the RADIUS protocol to address this fundamental limitation, and thus allowing for more efficient and more scalable implementations.

1. Introduction

The "Identifier" field in the RADIUS packet [RFC2865] is used to match outstanding requests and replies. As the field is one octet in size, only 256 requests can be in progress between two endpoints, which would present a significant bottleneck for performance. The workaround for this limitation is to use multiple source ports as documented and discussed in [RFC2865], [RFC3539], and [RFC6613].

Currently it is quite common to have hundreds of parallel connections between a RADIUS client and a server, especially in the deployment of controllers for wireless clients. As the scale requirement continues to increase, the number of "parallel connections" is expected to grow (perhaps reaching thousands), which will undoubtedly create a number of challenges with resource utilization, efficiency, and connection management (with RADIUS over TCP [RFC6613] in particular) on both the client and the server.

In this document we propose extensions to the RADIUS protocol to address this fundamental limitation and thus allowing for more efficient and more scalable implementations. More specifically, a new attribute ("Extended Identifier Attribute") is defined that can be used to discover the support of this specification between a client and a server using the Status-Server message [RFC5997]. Once the support is confirmed, the attribute can then be used to carry the extended identifier parameter in subsequent RADIUS packets. The extended identifier parameter can be used together with the Identifier to match outstanding requests and replies.

For brevity the extensions specified in this document are referred to as "the Extended Identifier feature" hereafter.

1.1. Specification of Requirements

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 [RFC2119].

the attribute list of a RADIUS packet. If the attribute does not appear as the first one in the attribute list of a RADIUS packet, the RADIUS packet MUST be treated as invalid and the packet be discarded according to [RFC2865].

Due to the hop-by-hop nature of RADIUS packet transmission between RADIUS devices, a PROXY server MUST strip the "Extended Identifier Attribute" (and reconstruct if appropriate) before sending the packet over a different session.

2.2. Status-Server Considerations

This section extends processing of Status-Server messages as described in Sections 4.1 and 4.2 of [RFC5997].

Prior to sending a RADIUS packet (other than the Status-Server request) with the "Extended Identifier Attribute", a client implementing this specification SHOULD first send a Status-Server request with the "Extended Identifier Attribute" to indicate its support for the Extended Identifier feature.

When a server implementing this specification receives a Status-Server request with the "Extended Identifier Attribute", it MUST include the "Extended Identifier Attribute" in its response to indicate whether it supports the Extended Identifier feature. If the Status-server reply from a server does not contain the "Extended Identifier Attribute", the client MUST treat this case as "reject" by the server for the Extended Identifier feature.

Unless specified by configuration, a client MUST NOT send a RADIUS packet (other than the Status-Server request) with the "Extended Identifier Attribute" to a server until it has received a response from the server confirming its support for the Extended Identifier feature using the "Extended Identifier Attribute".

When TCP is used as the transport protocol for RADIUS [RFC6613] between a client and a server, the Extended Identifier feature SHOULD be discovered each time the TCP session is established.

2.3. Use of the Extended Identifier Field

After the functionality defined in this specification is discovered between the client and the server, the Extended Identifier field can be carried using the "Extended Identifier Attribute" in a RADIUS packet. The Extended Identifier is to be used together with the Identifier to identify requests and replies. The assignment of these

parameters is left to implementation.

When the "Extended Identifier Attribute" is present in a RADIUS packet other than the Status-Server request or reply, the Extended Identifier field in the attribute MUST be used together with the Identifier field to identify requests and replies.

In response to a request from a client that contains the Extended Identifier field, the server MUST include the Extended Identifier field with an identical value in its reply.

3. IANA Considerations

A new attribute ("Extended Identifier Attribute") is defined for the RADIUS protocol. The type value [RFC3575] needs to be assigned using the assignment rules in section 10.3 of [RFC6929].

4. Security Considerations

This document defines a new RADIUS attribute, which does not affect the security considerations of the RADIUS protocol [RFC2865].

The new RADIUS attribute and the procedures described in this document helps eliminate the need for "parallel connections" between a RADIUS client and a server due to the limitation with the "Identifier" field. Thus the resource utilization (such as the number of UDP/TCP ports) on a RADIUS device is expected to be reduced significantly in large scale deployment.

5. Acknowledgments

We would like to thank Alan DeKok for useful discussions and suggestions.

6. References

6.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, 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.
- [RFC3575] Aboba, B., "IANA Considerations for RADIUS (Remote Authentication Dial In User Service)", RFC 3575, July 2003.

6.2. Informative References

- [RFC3539] Aboba, B. and J. Wood, "Authentication, Authorization and Accounting (AAA) Transport Profile", RFC 3539, June 2003.
- [RFC6613] DeKok, A., "RADIUS over TCP", RFC 6613, May 2012.
- [RFC5997] DeKok, A., "Use of Status-Server Packets in the Remote Authentication Dial In User Service (RADIUS) Protocol", RFC 5997, August 2010.
- [RFC6929] DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", RFC 6929, April 2013.

7. Authors' Addresses

Enke Chen
Cisco Systems
560 McCarthy Blvd.
Milpitas, CA 95035
USA

Email: enkechen@cisco.com

Naiming Shen
Cisco Systems

560 McCarthy Blvd.
Milpitas, CA 95035
USA

Email: naiming@cisco.com

Network Working Group
INTERNET-DRAFT
Updates: 5176
Category: Standards Track
<draft-dekok-radext-coa-proxy-00.txt>
3 July 2014

DeKok, Alan
FreeRADIUS
J. Korhonen
Nokia Siemens Networks

Dynamic Authorization Proxying in
Remote Authorization Dial-In User Service Protocol (RADIUS)
draft-dekok-radext-coa-proxy-00.txt

Abstract

RFC 5176 defines Change of Authorization (CoA) and Disconnect Message (DM) behavior for RADIUS. Section 3.1 of that document suggests that proxying these messages is possible, but gives no guidance as to how that is done. This specification corrects that omission.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 4, 2015.

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	4
1.1. Terminology	4
1.2. Requirements Language	4
2. Problem Statement	5
2.1. Typical RADIUS Proxying	5
2.2. CoA Processing	5
2.3. Failure of CoA Proxying	5
3. How to Perform CoA Proxying	6
3.1. Operator-NAS-Identifier	6
4. Functionality	7
4.1. User Login	7
4.2. CoA Proxing	8
5. Security Considerations	8
6. IANA Considerations	9
7. References	9
7.1. Normative References	9
7.2. Informative References	9

1. Introduction

RFC 5176 [RFC5176] defines Change of Authorization (CoA) and Disconnect Message (DM) behavior for RADIUS. Section 3.1 of that document suggests that proxying these messages is possible, but gives no guidance as to how that is done. This omission means that proxying of CoA packets is, in practice, impossible.

We correct that omission here.

1.1. Terminology

This document frequently uses the following terms:

Network Access Identifier

The Network Access Identifier (NAI) is the user identity submitted by the client during network access authentication. The purpose of the NAI is to identify the user as well as to assist in the routing of the authentication request. Please note that the NAI may not necessarily be the same as the user's email address or the user identity submitted in an application layer authentication.

Network Access Server

The Network Access Server (NAS) is the device that clients connect to in order to get access to the network. In PPTP terminology, this is referred to as the PPTP Access Concentrator (PAC), and in L2TP terminology, it is referred to as the L2TP Access Concentrator (LAC). In IEEE 802.11, it is referred to as an Access Point.

Home Network

The home network of a user.

Visited Network

The network which is accessed by a user, when that network is not their home network.

1.2. 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 [RFC2119].

2. Problem Statement

This section describes how RADIUS proxying works, how CoA packets work, and why CoA proxying does not work in the current system.

2.1. Typical RADIUS Proxying

When a RADIUS server proxies an Access-Request packet, it typically does so based on the contents of the User-Name attribute, which contains Network Access Identifier [NAI]. Other methods are possible, but we restrict ourselves to the most common usage.

The proxy server looks up the "Realm" portion of the NAI in a logical AAA routing table, as described in Section 3 of [NAI]. The entry in that table is the "next hop" to which the packet is sent. This "next hop" may be another proxy, or it may be the home server for that realm.

The "next hop" may perform the same Realm lookup, and the proxy the packet also. Alternatively, if the "next hop" is the Home Server for that realm, it will typically authenticate the user, and respond with an Access-Accept, Access-Reject, or Access-Challenge.

The response can be returned from the home server to the visited network, because each proxy server tracks the requests it has forwarded. When a response packet is by the proxy, it is matched to an incoming request, which lets the proxy forward the response to the source of the original request.

2.2. CoA Processing

[RFC5176] describes how CoA clients (often RADIUS servers) will send packets to CoA servers (often RADIUS clients). In typical use, CoA packets are sent within one network. That is, within the same "Realm". When used within one "Realm", there is only one "hop" for packets to take, so no proxying is necessary.

2.3. Failure of CoA Proxying

In the case of CoA proxying, the above scenarios fail. CoA packets may be sent minutes to hours after reception of the original Access-Request. In addition, the packet codes are different, so there is no way to match a CoA-Request packet to a particular Access-Request packet. There is therefore no "reverse path" for the CoA packet to follow.

As with Access-Request proxying, CoA proxying can be done done between Realms. There exists potentially multiple "hops" for packets

to follow. Packets cannot be forwarded to the Visited Network, based on the contents of the User-Name attribute, as that contains the Realm of the Home Network.

The conclusion is therefore that CoA proxying is impossible when using behavior defined in [RFC5176]. There is, however a solution.

3. How to Perform CoA Proxying

The solution is seen in the Operator-Name attribute defined in [RFC5580], Section 4.1. We repeat portions of that definition here for clarity:

This attribute carries the operator namespace identifier and the operator name. The operator name is combined with the namespace identifier to uniquely identify the owner of an access network.

Followed by a description of the REALM namespace:

REALM ('1' (0x31)):

The REALM operator namespace can be used to indicate operator names based on any registered domain name. Such names are required to be unique, and the rights to use a given realm name are obtained coincident with acquiring the rights to use a particular Fully Qualified Domain Name (FQDN). ...

In short, the Operator-Name attribute contains the an ASCII "1", followed by the Realm of the Visited Network. e.g. for the "example.com" realm, the Operator-Name attribute contains the text "1example.com". This information is precisely what we need to perform CoA proxying.

The only missing information is which NAS is managing the user. We may expect that the Visited Network will track this information, but there is no requirement for it to do so. We therefore need an additional attribute to contain this information.

3.1. Operator-NAS-Identifier

The Operator-NAS-Identifier attribute contains opaque information identifying a NAS. It MAY appear in the following packets: Access-Request, Accounting-Request, CoA-Request, DM-Request. Operator-NAS-Identifier MUST NOT appear in any other packet.

Operator-NAS-Identifier MAY occur in a packet if the packet also contains an Operator-Name attribute. Operator-NAS-Identifier MUST NOT appear in a packet if there is no Operator-Name in the packet.

Operator-NAS-Identifier MUST NOT occur more than once in a packet.

When an Operator-NAS-Identifier attribute is added by a proxy in a Visited Network, the following attributes MUST be deleted: NAS-IP-Address, NAS-IPv6-Address, NAS-Identifier. The proxy MUST then add a NAS-Identifier attribute, in order satisfy the requirements of Section 4.1 of [RFC2865], and of [RFC2866]. We suggest that the contents of the NAS-Identifier be the Realm name of the Visited Network.

Description

An opaque token describing the NAS a user has logged into.

Type

TBD. To be assigned by IANA

Length

TBD. Depends on IANA allocation.

Implementations supporting this attribute MUST be able to handle between one (1) and twenty (20) octets of data. Implementations creating an Operator-NAS-Identifier SHOULD NOT create attributes with more than twenty octets of data. A twenty octet string is more than sufficient to individually address all of the NASes on the planet.

Data Type

string. See [DATA] Section 2.6 for a definition.

Value

The contents of this attribute are an opaque token interpretable only by the Visited Network. The attribute MUST NOT contain any secret or private information.

4. Functionality

This section describes how the two attributes work together to permit CoA proxying.

4.1. User Login

The user logs in. When a Visited Network sees that the packet is proxied, it adds an Operator-Name with "1" followed by it's own realm

name. It MAY also add an Operator-NAS-Identifier.

The proxies then forward the packet. They MUST NOT delete or modify Operator-Name and/or Operator-NAS-Identifier.

The Home Server records both Operator-Name and Operator-NAS-Identifier along with other information about the users session.

4.2. CoA Proxying

When the Home Server decides to disconnect a user, it looks up the Operator-Name and Operator-NAS-Identifier, along with other user session identifiers as described in [RFC5176]. It then looks up the Operator-Name in the logical AAA routing table to find the CoA server for that realm (which may be a proxy). The CoA-Request is then sent to that server.

The CoA server receives the request, and if it is a proxy, performs a similar lookup as done by the Home Server. The packet is then proxied repeatedly until it reaches the Visited Network.

If the proxy cannot find a destination for the request, or if no Operator-Name attribute exists in the request, the proxy returns a CoA-NAK with Error-Cause 502 (Request Not Routable).

The Visited Network receives the CoA-Request packet, and uses the Operator-NAS-Identifier attribute to determine which local CoA server (i.e. NAS) the packet should be sent to.

If no CoA server can be found, the Visited Network return a CoA-NAK with Error-Cause 403 (NAS Identification Mismatch).

Any response from the CoA server (NAS) is returned to the Home Network.

5. Security Considerations

This specification incorporates by reference the [RFC6929] Section 11. In short, RADIUS has known issues which are discussed there.

This specification adds one new attribute, and defines new behavior for RADIUS proxying. As this behavior mirrors existing RADIUS proxying, we do not believe that it introduces any new security issues.

Operator-NAS-Identifier should remain secure. We don't say how.

6. IANA Considerations

IANA is instructed to allocated one new RADIUS attribute, as per Section 3.1, above.

7. References

7.1. Normative References

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", 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.

[RFC5580]

Tschofenig H., Ed. "Carrying Location Objects in RADIUS and Diameter", RFC 5580, August 2009.

[RFC6929]

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

[NAI]

DeKok A., "The Network Access Identifier", draft-ietf-radext-nai-06.txt, June 2013.

[DATA]

DeKok A., "Data Types in the Remote Authentication Dial-In User Service Protocol (RADIUS)", draft-dekok-radext-datatypes-04.txt, Juen 2014

7.2. Informative References

[RFC2866]

Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.

[RFC5176]

Chiba, M. et al, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January 2008.

Acknowledgments

Stuff

Authors' Addresses

Alan DeKok
The FreeRADIUS Server Project

Email: aland@freeradius.org

Jouni Korhonen
Nokia Siemens Networks
Linnoitustie 6
Espoo FI-02600
Finland

EMail: jouni.nospam@gmail.com

Network Working Group
INTERNET-DRAFT
Category: Proposed Standard
Updates: 2885
<draft-dekok-radext-request-authenticator-03.txt>
Expires: May 04, 2018
4 December 2017

Alan DeKok
FreeRADIUS

Correlating requests and replies in the
Remote Authentication Dial In User Service (RADIUS) Protocol
via the Request Authenticator.
draft-dekok-radext-request-authenticator-03.txt

Abstract

RADIUS uses a one-octet Identifier field to correlate requests and responses, which limits clients to 256 "in flight" packets per connection. This document removes that limitation by allowing Request Authenticator to be used as an additional field for identifying packets. The capability is negotiated on a per-connection basis, and requires no changes to the RADIUS packet format, attribute encoding, or data types.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on November 18, 2017.

Copyright Notice

Copyright (c) 2017 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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

11.1.	3
1. Introduction	4
1.1. Compatability with Existing RADIUS	5
1.2. Outline of the document	6
1.3. Terminology	6
1.4. Requirements Language	7
2. Review of Current Behavior	8
2.1. Client Behavior	8
2.2. Server Behavior	9
3. Alternative Approaches	10
3.1. Multiple Source Ports	11
3.2. Diameter	11
3.3. Multiple RADIUS packets in UDP	11
3.4. An Extended ID field	12
3.5. This Specification	13
4. Protocol Overview	13
4.1. Why this works	14
5. Signaling via Status-Server	15
5.1. Static Configuration	17
6. Original-Request-Authenticator Attribute	17
7. Transport Considerations	19
7.1. UDP	19
7.2. TCP	20
7.3. Dynamic Discovery	21
7.4. Connection Issues	21
8. System Considerations	22
8.1. Client Considerations	22
8.2. Server Considerations	23
8.3. Proxy Considerations	24
9. IANA Considerations	24
10. Security Considerations	25
10.1. Access-Request Forging	25
10.2. MD5 Collisions	26
11. Normative References	27
11.1.	27

1. Introduction

The Remote Authentication Dial In User Service (RADIUS) protocol contains an Identifier field, defined in [RFC2865] Section 5 as:

The Identifier field is one octet, and aids in matching requests and replies. The RADIUS server can detect a duplicate request if it has the same client source IP address and source UDP port and Identifier within a short span of time.

The small size of the field allows for only 256 outstanding requests without responses. If a client requires more than 256 packets to be outstanding to the RADIUS server, it must open a new connection, with a new source port.

This limitation does not severely impact low-load RADIUS systems. However, it is an issue for high-load systems. Opening new sockets is more expensive than tracking requests inside of an application, and is generally unnecessary in other UDP protocols.

For very high load systems, this "new socket" requirement can result in a client opening hundreds or thousands of connections. There are a number of problems with this approach:

- * RADIUS is connection oriented, and each connection operates independently of all other connections.
- * each connection created by the client must independently discover server availability. i.e. the connections can have different views of the status of the server, leading to packet loss and network instability.
- * The small size of RADIUS packets means that UDP traffic can reach Ethernet rate limits long before bandwidth limits are reached for the same network. This limitation prevents high-load systems from fully using available network bandwidth.
- * The limit of 256 outstanding requests means that RADIUS over TCP [RFC6613] is also limited to a small amount of traffic per connection, and thus will rarely achieve the full benefit of TCP transport.
- * the existence of hundreds of simultaneous connections can impose significant management overhead on clients and servers.
- * network stacks will generally operate more efficiently with a larger amount of data over one connection, instead of small amounts of

data
split over many connections.

For these reasons, it is beneficial to extend RADIUS to allow more than 256 outstanding requests per connection.

1.1. Compatability with Existing RADIUS

It is difficult in practice to extend RADIUS. Any proposal must not only explain why it cannot use Diameter, but it also must fit within the technical limitations of RADIUS.

We believe that this specification is appropriate for RADIUS, due to the following reasons:

- * this specification makes no change to the RADIUS packet format;
- * this specification makes no change to RADIUS security;
- * this specification adds no new RADIUS datatypes;
- * this specification uses standard RADIUS attribute formats;
- * this specification uses standard RADIUS data types;
- * this specification adds a single attribute to the RADIUS Attribute Type registry;
- * all existing RADIUS clients and servers will accept packets following this specification as valid RADIUS packets;
- * due to negotiation of capabilities, implementations of this specification are fully compatible with all existing RADIUS implementations;
- * clients implementing this specification can fall back to standard RADIUS in the event of misbehavior by a server;
- * servers implementing this specification use standard RADIUS unless this functionality has been explicitly negotiated;
- * low-load RADIUS systems do not need to implement this specification;
- * high-load systems can use this specification to remove all RADIUS-specific limits on filling available network bandwidth;
- * this specification allows for effectively unlimited numbers of

RADIUS packets over one connection, removing almost all issues related to connection management from client and server;

- * as a negative, implementations of this specification must have code paths for standard RADIUS, as well as for this specification.

In short, this specification is largely limited to changing the way that clients and server implementations internally match requests and responses.

We believe that the benefits of this specification outweigh the costs of implementing it.

1.2. Outline of the document

The document gives a high level overview of proposal. It then describes how the functionality is signaled in a Status-Server [RFC5997] It then defines the Original-Request-Authenticator attribute. It then describes how this change to RADIUS affects each type of packet (ordered by Code). Finally, it describes how the change affects transports such as UDP, TCP, and TLS.

1.3. Terminology

This document uses the following terms:

ID tracking

The traditional RADIUS method of tracking request / response packets by use of the Identifier field. Many implementations also use a socket descriptor, and/or src/dst IP/port as part of the tuple used to track packets.

ORA tracking

The method defined here which allows request / response packets extends ID tracking, in order to add Request Authenticator or Original-Request-Authenticator as part of the tuple used to track packets.

Network Access Server (NAS)

The device providing access to the network. Also known as the Authenticator (in IEEE 802.1X terminology) or RADIUS client.

RADIUS Proxy

In order to provide for the routing of RADIUS authentication and accounting requests, a RADIUS proxy can be employed. To the NAS, the RADIUS proxy appears to act as a RADIUS server, and to the RADIUS server, the proxy appears to act as a RADIUS client.

'request packet' or 'request'

One of the allowed packets sent by a client to a server. e.g. Access-Request, Accounting-Request, etc.

'response packet' or 'response'

One of the allowed packets sent by a server to a client, in response to a request packet. e.g. Access-Accept, Accounting-Response, etc.

1.4. Requirements Language

In this document, several words are used to signify the requirements of the specification. 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 [RFC2119].

2. Review of Current Behavior

In this section, we give a short review of how clients and servers currently operate. This review is necessary to help contextualize the subsequent discussion.

2.1. Client Behavior

When a client sends a request to a server, it allocates a unique Identifier for that packet, signs the packet, and sends it to the server over a particular network connection. When a client receives a response from a server over that same network connection, the client uses the Identifier to find the original request. The client then uses the original Request Authenticator to verify the Response Authenticator of the response.

We call this behavior "ID tracking". It is the traditional method used in RADIUS to track requests and responses. The term "ID tracking" is used in preference to "traditional RADIUS".

This tracking behavior is similar across all packet types. However, the management of the ID field is largely unspecified. For example, [RFC2865] Section 5 says

The Identifier field is one octet, and aids in matching requests and replies. The RADIUS server can detect a duplicate request if it has the same client source IP address and source UDP port and Identifier within a short span of time.

Similar text is contained in [RFC2866] Section 3, while [RFC5176] Section 2.3 has a more extensive discussion of the use of the Identifier field. However, all three documents are silent on the topic of how Identifiers are managed.

This silence means that there are no guidelines in the specifications for how a client can re-use an Identifier value. For example, can a client re-use an Identifier after a response is received? Can a client re-use an Identifier after a timeout, when no response has been received? If the client sends multiple requests and finally receives a response, can the Identifier be re-used immediately, or should the client wait until it receives all duplicate responses to its duplicate requests?

There are no clear answers to any of these questions.

The specifications are not much clearer on the subject of response packets. For example, [RFC2865] Section 4.2 has the following text for Access-Accept responses:

On reception of an Access-Accept, the Identifier field is matched with a pending Access-Request. The Response Authenticator field MUST contain the correct response for the pending Access-Request. Invalid packets are silently discarded.

[RFC2866] Section 4.2 has similar text, while [RFC5176] has no such text, and assumes that the reader knows that Identifiers are used to match a CoA-ACK packet to a CoA-Request packet.

While these issues are undefined, they nevertheless have to be addressed in practice. Client implementations have therefore chosen some custom behavior for Identifier management. This behavior has proven to be inter-operable, despite being poorly defined.

We bring this issue up solely to note that much of the RADIUS protocol is undefined or implementation-defined. As a result, it should be possible to define behavior which was previously left open to implementation interpretation. Even better, this new behavior can be defined in such a way as to obtain new and useful features in RADIUS.

2.2. Server Behavior

Server implementations have similar issues related to managing Identifiers for request packets. For example, while clients are permitted to send duplicate requests, [RFC2865] is not clear on how servers should handle those duplicates.

That issue was addressed in [RFC5080] Section 2.2.2, which defines how servers should perform duplicate detection. This duplicate detection aids in lowering the load on servers by allowing them to send cached responses to duplicates, instead of re-processing the request.

However, the issue of duplicate packets and retransmissions by the clients can result in another situation which is not discussed in any RADIUS specification. This topic deserves more discussion, because as we will see below, this topic motivates this specification.

The specifications do not describe what a server should do if it receives a new packet which is on the same connection as a previous one, and shares the Code and Identifier fields, but for which the Request Authenticator is different. That is, a client may send a request using an Identifier, not get a response, then time out that request. The client is then implicitly allowed by the specifications to re-use that Identifier when sending a new request.

That new request will be on the same connection as a previous

request, using the same Code and Identifiers as a previous request, but will have a different Request Authenticator.

When the server receives this new packet, it has no knowledge that the client has timed out the original request. The server may still be processing the original request. If the server responds to the original request, the response will likely get ignored by the client, as it has timed out that original request. If the server responds to the new request, the client will probably accept the response, but the server must then not respond to the original request.

The server now has two "live" requests from a client, but it can respond only to one. These request are "conflicting packets", and the process used to detect them is conflict detection and conflict management.

While the concept of "conflicting packets" is not defined in the RADIUS specifications, it nevertheless has had to be addressed in practice. Server implementations have each chosen some behavior for conflict detection and management. This implementation-defined behavior has proven to be inter-operable in practice, which allows RADIUS to operate in the face of conflicts.

The concept conflict detection is the key concept driving this specification. If servers were instead allowed to process and reply to both requests, then the limits of the Identifier field would be entirely bypassed.

The rest of this specification describes the impact of allowing these otherwise "conflicting packets" to be processed. We discuss a framework required to negotiate this functionality in an inter-operable manner. We define a new method of tracking packets, called "ORA tracking", which is discussed further below.

3. Alternative Approaches

There are other ways that the per-connection Identifier limitation could have been addressed. As extending RADIUS is a controversial topic, we believe it is useful to discuss some alternative approaches, and to explain their costs and benefits. This discussion ensures that readers of this specification are fully informed as to why this design is the preferred approach.

We finish this section by explaining why this solution was chosen.

3.1. Multiple Source Ports

One approach is to follow the suggestion of [RFC2865] Section 5, which says:

NAS needs more than 256 IDs for outstanding requests, it MAY use additional source ports to send requests from, and keep track of IDs for each source port. This allows up to 16 million or so outstanding requests at one time to a single server

This suggestion has been widely implemented. However, practice shows that the number of open ports has a practical limitation much lower than 65535. This limitation is due both to ports being used by other applications on the same system, and application and OS-specific complexities related to opening thousands of ports.

The "multiple source port" approach is workable for low-load systems. However, implementors of high-load systems have requested an alternative to that approach. The justification is that this approach has proven to be problematic in practice for them.

3.2. Diameter

Another common approach is to suggest that implementors switch to using Diameter instead of RADIUS. We believe that the Diameter protocol does not satisfy the requirements of the implementors who are requesting extensions to RADIUS.

The short summary is that implementors have significant investment in a RADIUS infrastructure. Switching to Diameter would involve either deprecating or throwing away all of that infrastructure. This approach is simply not technically or economically feasible.

Ad hoc surveys indicate that the majority of the implementations that will use this specification do not have pre-existing Diameter code bases. We suggest that it is simpler for implementors to make minor changes to their RADIUS systems than it is to implement a full Diameter stack. This reality is a major consideration when creating new specifications.

For these reasons, switching to Diameter is not a useful approach.

3.3. Multiple RADIUS packets in UDP

Another approach would be to allow multiple RADIUS packets in one UDP packet. This method would allow an increased amount of RADIUS traffic to be sent in the same number of UDP packets.

However, this approach still has the limit of 256 outstanding requests, which means that implementations have extra work of juggling RADIUS packets, UDP packets, and UDP connections. In addition, this approach does not address the issues discussed above for RADIUS over TCP.

As a result, this approach is not suitable as a solution to the ID limit problem.

3.4. An Extended ID field

Another approach is to use an attribute which contained an "extended" ID, typically one which is 32 bits in size. That approach has been used in at least one commercial RADIUS server for years, via a Vendor-Specific Attribute.

The benefits of this approach is that it makes no change to the RADIUS protocol format, attribute encoding, data types, security, etc. It just adds one "integer" attribute, as an "extended ID" field. Client implementations add the "extended ID" attribute to requests, and server implementations echo it back in responses. The "extended ID" field is also used in doing duplicate detection, finding conflicting packets, etc.

Implementations using this approach have generally chosen to not perform negotiation of this functionality. Instead, they require both client and server to be statically configured to enable the "extended ID".

Clients implementing the "extended ID" must, of course, manage this new identifier. As the identifier is still local to a connection, it is possible to simply take the "extended ID" from a counter which is incremented for every request packet. There is no need to mark these identifiers as unused, as the 32-bit counter space is enough to ensure that re-use happens rarely. i.e. at 1 million packets per second, the counter is enough to last for over an hour, which is a time frame much larger than the lifetime of any individual packet.

This approach is compatible with all RADIUS transports, which is a major point in its favor.

An implementation of the "extended ID" approach is less than 200 lines of C code [PATCH]. That patch includes capability negotiation via Status-Server, and full client / server / proxy support.

This approach has therefore been proven to be workable in practice, and simple to implement.

3.5. This Specification

The approach discussed here was chosen after looking at the handling of packet conflicts, as discussed above in Section X. The conclusion was that since the behavior around "conflicting packets" was entirely implementation-defined, then changing the behavior would involve minor changes to the RADIUS specifications.

This specification suggests that clients and servers can choose to use the Request Authenticator as an additional field to uniquely identify request packets. This choice is entirely local to the client or server implementation, and involves no changes to the wire format or wire protocol. There are additional considerations which are outlined below.

The approach outlined in this specification is largely similar to the "extended ID" approach, except that it leverages a pre-existing field as an identifier, instead of creating a new one. This re-use means that the client does not need to track or allocate new identifiers

The use of the Request Authenticator as an identifier means that there are 128 bits of space available for identifiers. This large space means that the "conflicting packet" problem is avoided almost entirely, as the Request Authenticator is either random data (Access-Request), or an MD5 signature (other packets).

The subject of MD5 collisions is addressed below, in Section X.

As with the "extended ID" approach, this approach is compatible with all transports.

We believe that this approach is slightly simpler than the next best approach ("extended ID"), while at the same time providing more benefits. As a result, it is the recommended approach for allowing more than 256 outstanding packets on one connection.

4. Protocol Overview

We extend RADIUS to allow the use of the Request Authenticator field as an additional identifier. Subject to certain caveats outlined below, the Request Authenticator can be treated as being temporally and globally unique. This uniqueness is what makes it a useful identifier field.

The capability is first negotiated via Status-Server, as outlined below. Once both client and server agree on the use of this capability, the client can start sending normal request packets.

The client creates a request packet as usual. When the client needs to store the request packet, it adds the Request Authenticator as part of the unique key which tracks the request. The packet is then sent to the server.

The server receives the request, and uses the Request Authenticator to track the request, in addition to any previously used information. When the server sends a reply, it copies the Request Authenticator to the response packet, and places the 16 octet value into the Original-Request-Authenticator attribute. The response is then sent as before.

When the client receives the response, it uses the Original-Request-Authenticator attribute to create the lookup key, in order to find the original request. Once the original request is found, packet verification and processing continues as before.

We note again that the Identifier field is still used to correlate requests and responses, along with any other information that the implementation deems necessary. (e.g. Code, socket descriptor, src/dst IP/port, etc.) The only change is that the Request Authenticator is added to that tuple.

In short, the "ID tracking" method of tracking requests and responses is extended to allow the use of the Request Authenticator as part of the tuple used to track requests and responses. This new tracking method is called "ORA tracking".

Further details and implementation considerations are provided below.

Since the Request Authenticator field is sixteen octets in size, this process allows an essentially unlimited number of requests to be outstanding on any one connection. This capability allows clients to open only one connection to a server, and to send all data over that connection. As noted above, using fewer connections will increase the clients ability to perform dead server detection, do fail-over, and will result in increased network stability.

4.1. Why this works

In this section, we explain why the Request Authenticator makes a good packet identifier.

For Access-Request packets, [RFC2865] Section 3 defines the Request Authenticator to contain random values. Further, it is suggested that these values "SHOULD exhibit global and temporal uniqueness". The same definition is used in [RFC5997] Section 3, for Status-Server packets. Experience shows that implementations follow this

suggestion. As a result, the Request Authenticator is a good identifier which uniquely identifies packets.

Other request packets create the Request Authenticator as an MD5 calculation over the packet and shared secret. i.e. MD5(packet + secret). The MD5 digest algorithm was designed to be strongly dependent on input data, and to have half of the output bits change if one bit changed in the input. As a result, the Request Authenticator is a good hash which can distinguish different packets.

The question is whether or not the Request Authenticator is a good identifier. The following discussion make this case.

One argument is that MD5 has low collision rates. In the absence of an explicit attack, there should be one collision every 2^{64} packets. Since the packet lifetime is small (typically 30 seconds maximum), we can expect a collision only if more than 2^{59} packets are sent during that time frame. For more typical use-cases, the packet rate is low enough (i.e. even 2^{20} packets per second), that there is a one in 2^{39} chance of a collision every 30 seconds.

We believe that such collision rates are acceptably low. Explicit attacks are discussed in the Security Considerations section, below.

The one case where collisions will occur naturally is when the packet contents are identical. For example, a transmission of a second Access-Request after the first one times out. In this situation, though, there is in fact no collision, as the input data is identical. Both requests are entirely identical, and any response to one is a response to both.

For non-identical requests, the packet typically contains the Identifier and Length fields, along with counters, timestamps, etc. These values change on a per-packet basis, making the Request Authenticator also change.

As a result, the MD5 signature of a request is appropriate to use as a packet identifier. In all cases (barring attacks), it will contain a globally and temporally unique identifier for the request.

5. Signaling via Status-Server

When a client supports this functionality, it sends a Status-Server packet to the server, containing an Original-Request-Authenticator attribute. See Section X, below, for the definition of the Original-Request-Authenticator attribute.

The contents of the Original-Request-Authenticator attribute in the

Status-Server packet MUST be zero. The Original-Request-Authenticator attribute MUST NOT appear in any request packet other than Status-Server. If a server does see this attribute in a request packet, the attribute MUST be treated as an "invalid attribute", and ignored as per [RFC6929] Section 2.8.

A server which supports this functionality SHOULD signal that capability to the client by sending a response packet which contains an Original-Request-Authenticator attribute. That attribute MUST contain an identical copy of the Request Authenticator from the original Status-Server packet.

When a client sends an Original-Request-Authenticator attribute in a Status-Server and does not receive that attribute in the response, the client MUST NOT use "ORA tracking" for requests and responses. The client MUST then behave as a normal RADIUS client, and use "ID tracking" for requests and response.

If a server does not support this functionality, it MUST NOT place an Original-Request-Authenticator attribute in the response packet. As the default behavior of existing RADIUS servers is to not place this attribute in the response to a Status-Server, negotiation will downgrade automatically to traditional RADIUS, and "ID tracking".

As the response to a Status-Server can use one of many RADIUS Codes, we use a generic "response" name above. See following sections for how to handle specific types of responses.

We note that "ORA tracking" negotiation SHOULD be done per connection. i.e. per combination of (transport, src/dst ip/port). Section X, below, discusses additional issues related to client/server connections. In this section, when we refer to a client and server performing negotiation, we mean that negotiation to be specific to a particular connection.

Once the "ORA tracking" has been negotiated on a connect, then all packets for that connection MUST use it, no matter what values they allow for the Code field. For example, [RFC6613] permits multiple Codes on one connection.

Even if a client and server negotiate "ORA tracking", the client can still fall back to "ID tracking". There is no need for the client to signal the server that this change has happened. The client can use "ID tracking" while the server uses "ORA tracking", as the two systems are entirely compatible from the client side.

The situation is a bit different for a server. Once "ORA tracking" has been negotiated, a server MUST use that method, and MUST include

the Original-Request-Authenticator attribute in all response packets. If a client negotiates "ORA tracking" on a connection and later sees response packets which do not contain an Original-Request-Authenticator attribute, the client SHOULD discard those non-compliant packets. For connection-oriented protocols, the client SHOULD close the connection.

There is a time frame during this failure process during which outstanding requests on that connection may not receive responses. This situation will result in packet loss, which will be corrected once the new connection is used. The possibility of such problems should be used instead by implementors as incentive to ensure that they do not create invalid Original-Request-Authenticator attributes. Implementing the specification correctly will prevent this packet loss from occurring.

The negotiation outlined here ensures that RADIUS clients and servers supporting this functionality are entirely backwards compatible with existing RADIUS clients and servers.

5.1. Static Configuration

As an alternative to using Status-Server, clients and servers MAY be administratively configured with a flag which indicates that the other party supports this functionality. Such a flag can be used where the parties are known to each other. Such a flag is not appropriate for dynamic peer discovery [RFC7585], as there are no provisions for encoding the flag in the DNS queries or responses.

When a client is administratively configured to know that a server supports this functionality, it SHOULD NOT do negotiation via Status-Server.

If a client is administratively configured to believe that a server supports the Original-Request-Authenticator attribute, but the response packets do not contain an Original-Request-Authenticator attribute, the client MUST update its configuration to mark the server as not supporting this functionality.

This process allows for relatively simple downgrade negotiation in the event of misconfiguration on either the client or the server.

6. Original-Request-Authenticator Attribute

We define a new attribute, called Original-Request-Authenticator. It is intended to be used in response packets, where it contains an exact copy of the Request Authenticator field from the original request that elicited the response.

As per the suggestions of [RFC8044], we describe the attribute using a data type defined therein, and without the use of ASCII art.

Type

TBD - IANA allocation from the "extended" type space

Length

19 - TBD double-check this after IANA allocation

Data Type

octets

Value

MUST be 16 octets in length. For Status-Server packets, the contents of the Value field MUST be zero. For response packets, the contents of the Value field MUST be a copy of the Request Authenticator from the original packet that elicited the response.

The Original-Request-Authenticator attribute can be used in a Status-Server packet.

The Original-Request-Authenticator attribute can be used in a response packet. For example, it can be used in an Access-Accept, Accounting-Response, CoA-ACK, CoA-NAK, etc.

Note that this document updates multiple previous specifications, in order to allow this attribute in responses.

- * [RFC2865] Section 5.44 is updated to allow Original-Request-Authenticator in Access-Accept, Access-Reject, and Access-Challenge responses
- * [RFC2866] Section 5.13 is updated to allow Original-Request-Authenticator in Accounting-Response responses.
- * [RFC5176] Section 3.6 is updated to allow Original-Request-Authenticator in CoA-ACK, CoA-NAK, Disconnect-Ack, and Disconnect-NAK responses.

The Original-Request-Authenticator attribute MUST NOT be used in any request packet. That is, it MUST NOT be used in an Access-Request, Accounting-Request, CoA-Request, or Disconnect-Request packets.

When it is permitted in a packet, the Original-Request-Authenticator

attribute MUST exist either zero or one times in that packet. There MUST NOT be multiple occurrences of the attribute in a packet.

The contents of the Original-Request-Authenticator attribute MUST be an exact copy of the Request Authenticator field of a request packet sent by a client. As with "ID tracking", the Identifier field in the response MUST match the Identifier field in a request.

Where the format and/or contents of the Original-Request-Authenticator attribute does not meet these criteria, the received attribute MUST be treated as an "invalid attribute" as per [RFC6929], Section 2.8. That is, when an invalid Original-Request-Authenticator attribute is seen by either a client or server, their behavior is to behave as if the attribute did not exist.

7. Transport Considerations

This section describes transport considerations for this specification.

The considerations for DTLS are largely the same as for UDP. The considerations for TLS are largely the same as for TCP. We therefore do not have different sections herein for the TLS-enabled portion of the protocols.

7.1. UDP

RADIUS over UDP is defined in [RFC2866]. RADIUS over DTLS is defined in [RFC7360].

When negotiated by both peers, this proposal changes the number of requests which can be outstanding over a UDP connection.

Where clients are sending RADIUS packets over UDP, they SHOULD include the Original-Request-Authenticator attribute in all Status-Server messages to a server, even if the functionality has been previously negotiated. While the client can generally assume that a continual flow of packets means that the server has not been changed, this assumption is not true when the server is unresponsive, and the client decides it needs to send Status-Server packets.

Similarly, the server cannot assume that it is respond to the same client on every packet. However, once Original-Request-Authenticator has been negotiated, the server can safely include that attribute in all response packets to that client. If the client changes to not supporting the attribute, the attribute will be ignored by the client, and the behavior falls back to standard RADIUS.

Where clients are sending RADIUS packets over DTLS, there is an underlying TLS session context. The client can therefore assume that all packets for one TLS session are for the same server, with the same capabilities. The server can make the same assumption.

7.2. TCP

RADIUS over TCP is defined in [RFC6614]. RADIUS over TLS is defined in [RFC6614].

When negotiated by both peers, this proposal changes the number of requests which can be outstanding over a TCP connection.

Status-Server packets are also used by the application-layer watchdog, described in [RFC6614] Section 2.6. Where clients have previously negotiated Original-Request-Authenticator for a connection, they MUST continue to send that attribute in all Status-Server packets over that connection.

There are other considerations with the use of Status-Server. Due to the limitations of the ID field, [RFC6613] Section 2.6.5 suggests:

Implementations SHOULD reserve ID zero (0) on each TCP connection for Status-Server packets. This value was picked arbitrarily, as there is no reason to choose any one value over another for this use.

This restriction can now be relaxed when both client and server have negotiated the use of the Original-Request-Authenticator attribute. Or, with no loss of generality, implementations can continue to use a fixed ID field for Status-Server application watchdog messages.

We also note that the next paragraph of [RFC6614] Section 2.6.5. says:

Implementors may be tempted to extend RADIUS to permit more than 256 outstanding packets on one connection. However, doing so is a violation of a fundamental part of the protocol and MUST NOT be done. Making that extension here is outside of the scope of this specification.

This specification extends RADIUS in a standard way, making that recommendation redundant.

[RFC6613] Section 2.5 describes congestion control issues which affect inter-transport proxies. If both inbound and outbound transports support this specification, those congestion issues no longer apply.

If however, a proxy supports this specification on inbound connections but does not support it on outbound connections, then congestion may occur. The only solution here is to ensure that the proxy is capable of opening multiple source ports, as per [RFC2865] Section 5.

7.3. Dynamic Discovery

The dynamic discovery of RADIUS servers is defined in [RFC7585].

This specification is compatible with [RFC7585], with the exception of the statically configured flag described in Section X, above. As the server is dynamically discovered, it is impossible to have a static flag describing the server capabilities.

The other considerations for dynamic discovery are the same as for RADIUS over TLS.

7.4. Connection Issues

Where clients start a new connection to a server (no matter what the transport), they SHOULD negotiate this functionality for the new connection, unless the ability has been statically configured. There is no guarantee that the new connection goes to the same server.

When a client has zero connections to a server, it MUST perform this negotiation for the new connection, prior to using this functionality, unless the ability has been statically configured. There is every reason to believe that server has remained the same over extended periods of time.

If a client has one or more connections open to a server, and wishes to open a new one, it may skip the renegotiation.

Each client and server MUST negotiate and track this capability on a per-connection basis. Implementations MUST be able to send packets to the same peer at the same time, using both this method, and the traditional RADIUS ID allocation.

A client may have a backlog of packets to send while negotiating this functionality. In the interests of efficiency, it SHOULD send packets from that backlog while negotiation is taking place. As negotiation has not finished, these packets and their responses MUST be managed as per standard RADIUS.

After this functionality has been negotiated, new packets from that connection MUST follow this specification. Responses to earlier packets sent on that connection during the negotiation phase MUST be

accepted and processed.

We recognize that this tracking may be complex, which is why this behavior is not mandatory. Clients may choose instead to wait until negotiation is complete before sending packets; or to assume that the functionality of the server is the same across all connections to it, and therefore only do negotiations once.

8. System Considerations

This section describes implementation considerations for clients and servers.

8.1. Client Considerations

Clients SHOULD have an configuration flag which lets administrators statically configure this behavior for a server. Clients MUST otherwise negotiate this functionality before using it.

If this functionality has been negotiated, clients MUST use the Request Authenticator as an part of the Key used to uniquely identify request packets. Clients MUST use the Original-Request-Authenticator attribute from response packets as part of the Key to find the original request packet.

The Original-Request-Authenticator attribute has been (or is likely to be) allocated from the "extended" attribute space. We note that despite this allocation, clients are not required to implement the full [RFC6929] specification. That is, clients may be able to originate and receive Original-Request-Authenticator attributes, while still being unable to originate or receive any other attribute in the "extended" attribute space.

The traditional behavior of clients is to track one or more connections, each of which has 256 IDs available for use. As requests are sent, IDs are marked "used". As responses are received, IDs are marked "free". IDs may also marked "free" when a request times out, and the client gives up on receiving a response.

If all of the IDs for a particular connection are marked "free", the client opens a new connection, as per the suggestion of [RFC2865] Section 5. This connection and any associated IDs are then made available for use by new requests.

Similarly, when a client notices that all of the IDs for a connection are marked "free", it may close that connection, and remove the IDs from the ones available for use by new requests. The connections may have associated idle timeouts, maximum lifetimes, etc. to avoid

"connection flapping".

All of this management is complex, and can be expensive for client implementations. While this management is still necessary for backwards compatibility, this specification allows for a significantly simpler process for ID allocation. There is no need for the client to open multiple connections. Instead, all traffic can be sent over one connection.

In addition, there is no need to track "used" or "free" status for individual IDs. Instead, the client can re-use IDs at will, and can rely on the uniqueness of the Request Authenticator to disambiguate packets.

As there is no need to track ID status, the client may simply allocate IDs by incrementing a local counter.

With this specification, the client still needs to track all outgoing requests, but that work was already required in traditional RADIUS.

Client implementors may be tempted to require that the Original-Request-Authenticator be the first attribute after the RADIUS header. We state instead that clients implementing this specification **MUST** accept the Original-Request-Authenticator attribute, no matter where it is in the response. We remind implementors that this specification adds a new attribute, it does not change the RADIUS header.

Finally, we note that clients **MUST NOT** set the ID field to a fixed value for all packets. While it is beneficial to use the Request Authenticator as an identifier, removing the utility of an existing identifier is unwarranted.

8.2. Server Considerations

Servers **SHOULD** have an configuration flag which lets administrators statically configure this behavior for a client. Servers **MUST** otherwise negotiate this functionality before using it.

If this functionality has been negotiated, servers **MUST** use the Request Authenticator as an part of the key used to uniquely identify request packets. Servers **MUST** use the Original-Request-Authenticator attribute from response packets as part of the Key to find the original request packet.

The Original-Request-Authenticator attribute has been (or is likely to be) allocated from the "extended" attribute space. We note that despite this allocation, servers are not required to implement the

full [RFC6929] specification. That is, servers may be able to originate and receive Original-Request-Authenticator attributes, while still being unable to originate or receive any other attribute in the "extended" attribute space.

8.3. Proxy Considerations

There are additional considerations specific to proxies. [RFC6929] Section 5.2 says in part;

Proxy servers SHOULD forward attributes, even attributes that they do not understand or that are not in a local dictionary. When forwarded, these attributes SHOULD be sent verbatim, with no modifications or changes. This requirement includes "invalid attributes", as there may be some other system in the network that understands them.

On its face, this recommendation applies to the Original-Request-Authenticator attribute. The caveat is that Section X, above, requires that servers do not send the Original-Request-Authenticator to clients unless the clients have first negotiated the use of that attribute. This requirement should ensure that proxies which are unaware of the Original-Request-Authenticator attribute will never receive it.

However, if a server has been administratively configured to send Original-Request-Authenticator to a client, that configuration may be in error. In which case a proxy or originating client may erroneously receive that attribute. If the proxy or server is unaware of Original-Request-Authenticator, then no harm is done.

It is possible for a proxy or client to be aware of Original-Request-Authenticator, and not negotiate it with a server, but that server (due to issues outlined above) still forwards the attribute to the proxy or client. In that case, the requirements of Section X, above, are that the client treat the received Original-Request-Authenticator attribute as an "invalid attribute", and ignore it.

The net effect of these requirements and cross-checks is that there are no interoperability issues between existing RADIUS implementations, and implementations of this specification.

9. IANA Considerations

This specification allocates one attribute in the RADIUS Attribute Type registry, as follows.

Name

Original-Request-Authenticator

Type

TBD - allocate from the "extended" space

Data Type

octets

10. Security Considerations

This proposal does not change the underlying RADIUS security model, which is poor.

The contents of the Original-Request-Authenticator attribute are the Request Authenticator, which is already public information for UDP or TCP transports.

The use of Original-Request-Authenticator is defined in such a way that all systems fall back gracefully to using standard RADIUS. As such, there are no interoperability issues between this specification and existing RADIUS implementations.

There are few, if any, security considerations related to implementations. Clients already must track the Request Authenticator, so matching it in a response packet is minimal extra work. Servers must also track and cache duplicate packets, as per [RFC5080] Section 2.2.2, so using the Request Authenticator as an additional identifier is minimal extra work.

The use (or not) of Original-Request-Authenticator has no other security considerations, as it is used solely as an identifier to match requests and responses. It has no other meaning or use.

10.1. Access-Request Forging

The Request Authenticator in Access-Request packets is defined to be a 16 octet random number [RFC 2865] Section 3. As such, these packets can be trivially forged.

The Message-Authenticator attribute was defined in [RFC2869] Section 5.14 in order to address this issue. Further, [RFC5080] Section 2.2.2 suggests that client implementations SHOULD include a Message-Authenticator attribute in every Access-Request to further help mitigate this issue.

The Status-Server packets also have a Request Authenticator which is a 16-octet random number [RFC5997] Section 3. However, [RFC5997]

Section 2 says that a Message-Authenticator attribute MUST be included in every Status-Server packet, which provides per-packet authentication and integrity protection.

We extend that suggestion for this specification. Where the transport does not provide for authentication or integrity protection (e.g. RADIUS over UDP or RADIUS over TCP), each Access-Request packet using this specification MUST include a Message-Authenticator attribute. This inclusion ensures that packets are accepted only from clients which know the RADIUS shared secret.

This protection is, of course, insufficient. Malicious or misbehaving clients can create Access-Request packets which re-use Request Authenticators. These clients can also create Request Authenticators which exploit implementation issues in servers, such as turning a simply binary lookup into a linked list lookup.

As a result, server implementations MUST NOT assume that the Request Authenticator is random. Server implementations MUST be able to detect re-use of Request Authenticators.

When a server detects that a Request Authenticator is re-used, it MUST replace the older request with the newer request. It MUST NOT respond to the older request. It SHOULD issue a warning message to the administrator that the client is malicious or misbehaving.

Server implementations SHOULD use data structures such as Red-Black trees, which are immune to maliciously crafted Request Authenticators.

10.2. MD5 Collisions

For other packet types (Accounting-Request, etc.), the Request Authenticator is the MD5 signature of the packet and the shared secret. Since this data is used directly as an identifier, we need to examine the security issues related to this practice.

We must note that MD5 has been broken, in that there is a published set of work which describes how to create two sets of input data which have the same MD5 hash. These attacks have been extended to create sets of data of arbitrary length, which differ only in 128 bytes, and have the same MD5 hash.

This attack is possible in RADIUS, as the protocol has the capability to transport opaque binary data in (for example) Vendor-Specific attributes. There is no need for the client or server to understand the data, it simply has to exist in the packet for the attack to succeed.

Another attack allows two sets of data to have the same MD5 hash, by appending thousands of bytes of carefully crafted data to the end of the file. This attack is also possible in RADIUS, as the maximum packet size for UDP is 4096 octets, and [RFC7930] permits packets up to 65535 octets in length.

However, as the packets are authenticated with the shared secret, these attacks can only be performed by clients who are in possession of the shared secret. That is, only trusted clients can create MD5 collisions.

We note that this specification requires server implementations to detect duplicates, and to process only one of the packets. This requirement could be exploited by a client to force a server to do large amounts of work, partially processing a packet which is then made obsolete by a subsequent packet. This attack can be done in RADIUS today, so this specification adds no new security issues to the protocol.

In fact, this specification describes the problem of "conflicting packets" for the first time, and defines how they should be processed by servers. This addition to the RADIUS protocol in fact increases it's security, by specifying how this corner case should be handled. The fact that RADIUS has been widely implemented for almost 25 years without this issue being described shows that the protocol and implementations are robust.

We do not offer a technical solution to the problem of trusted parties misbehaving. Instead, the problem should be noted by the server which is being attacked, and administrative (i.e. human) intervention should take place.

11. Normative References

11.1.

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", 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.

[RFC2869]

Rigney, C. et. al., "RADIUS Extensions", RFC 2869, June 2000.

[RFC5080]

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

[RFC6929]

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

[RFC8044]

DeKok, A., "Data Types in RADIUS", RFC 8044, January 2017.
Informative references

[RFC2866]

Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.

[RFC2866]

Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.

[RFC5176]

Chiba, M., et al, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January 2008.

[RFC5997]

DeKok, A., "Use of Status-Server Packets in the Remote Authentication Dial In User Service (RADIUS) Protocol", RFC 5997, August 2017.

[RFC6613]

DeKok, A., "RADIUS over TCP", RFC 6613, May 2012.

[RFC6614]

Winter, S., et al, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, May 2012.

[RFC7360]

DeKok, A., "Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS", RFC 7360, September 2014.

[RFC7585]

Winter, S., and McCauley, M., "Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)", RFC 7585, October 2015

[RFC7930]

Hartman, S., "Larger Packets for RADIUS over TCP", RFC 7930, August 2016.

[PATCH]

Naiming Shen, "Re: [radext] I-D Action: draft-chen-radext-
identifier-attr-01.txt",
<https://mailarchive.ietf.org/arch/msg/radext/BkXgD68MASxqD4vjXV1M2CaRWAY>,
April 2017.

Acknowledgments

Author's Address

Alan DeKok
The FreeRADIUS Server Project
<http://freeradius.org>
Email: aland@freeradius.org

.nr HY 0

Network Working Group
INTERNET-DRAFT
Updates: 5176, 5580
Category: Standards Track
<draft-ietf-radext-coa-proxy-10.txt>
22 January 2019

DeKok, Alan
FreeRADIUS
J. Korhonen

Dynamic Authorization Proxying in
Remote Authorization Dial-In User Service Protocol (RADIUS)
draft-ietf-radext-coa-proxy-10.txt

Abstract

RFC 5176 defines Change of Authorization (CoA) and Disconnect Message (DM) behavior for RADIUS. That document suggests that proxying these messages is possible, but gives no guidance as to how it is done. This specification updates RFC 5176 to correct that omission for scenarios where networks use Realm-based proxying as defined in RFC 7542. This specification also updates RFC 5580 to allow the Operator-Name attribute in CoA-Request and Disconnect-Request packets.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on July 22, 2019.

Copyright Notice

Copyright (c) 2019 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	4
1.1. Terminology	4
1.2. Requirements Language	5
2. Problem Statement	6
2.1. Typical RADIUS Proxying	6
2.2. CoA Processing	7
2.3. Failure of CoA Proxying	7
3. How to Perform CoA Proxying	8
3.1. Changes to Access-Request and Accounting-Request pack	9
3.2. Proxying of CoA-Request and Disconnect-Request packet	9
3.3. Reception of CoA-Request and Disconnect-Request packe	10
3.4. Operator-NAS-Identifier	11
4. Requirements	14
4.1. Requirements on Home Servers	14
4.2. Requirements on Visited Networks	14
4.3. Requirements on Proxies	15
4.3.1. Security Requirements on Proxies	15
4.3.2. Filtering Requirements on Proxies	16
5. Functionality	17
5.1. User Login	17
5.2. CoA Proxying	17
6. Security Considerations	18
6.1. RADIUS Security and Proxies	19
6.2. Security of the Operator-NAS-Identifier Attribute ...	19
7. IANA Considerations	20
8. References	20
8.1. Normative References	20
8.2. Informative References	21

1. Introduction

RFC 5176 [RFC5176] defines Change of Authorization (CoA) and Disconnect Message (DM) behavior for RADIUS. Section 3.1 of [RFC5176] suggests that proxying these messages is possible, but gives no guidance as to how that is done. This omission means that in practice, proxying of CoA packets is impossible.

We partially correct that omission here by explaining how proxying of these packets can be done by leveraging an existing RADIUS attribute, Operator-Name (Section 4.1 of [RFC5580]). We then explain how this attribute can be used by proxies to route packets "backwards" through a RADIUS proxy chain from a Home Network to a Visited Network. We then introduce a new attribute; Operator-NAS-Identifier. This attribute permits packets to be routed from the RADIUS server at the Visited Network to the NAS.

This correction is limited to the use-case of Realm-based proxying as defined in [RFC7542]. Other forms of proxying are possible, but are not discussed here. We note that the recommendations of this document apply only to those systems which implement proxying of CoA packets, and then only to those that implement Realm-based CoA proxying. This specification neither requires nor suggests changes to any implementation or deployment of any other RADIUS systems.

We also update the behavior of [RFC5580] to allow the Operator-Name attribute to be used in CoA-Request and Disconnect-Request packets, as further described in this document.

This document is a Proposed Standard in order to update the behavior of [RFC5580], which is also a Proposed Standard. This document relies heavily upon and also updates some behavior of RFC 5176, which is an Informational document; though the applicability statements in Section 1.1 of [RFC5176] do not apply to this document, this document does not change the status of [RFC5176].

We finally conclude with a discussion of the security implications of this design, and show that they do not decrease the security of the network.

1.1. Terminology

This document frequently uses the following terms:

CoA

Change of Authorization, e.g. CoA-Request, or CoA-ACK, or CoA-NAK, as defined in [RFC5176]. That specification also defines

Disconnect-Request, Disconnect-ACK, and Disconnect-NAK. For simplicity here, where we use "CoA", we mean a generic "CoA-Request or Disconnect-Request" packet. We use "CoA-Request" or "Disconnect-Request" to refer to the specific packet types.

Network Access Identifier

The Network Access Identifier (NAI) [RFC7542] is the user identity submitted by the client during network access authentication. The purpose of the NAI is to identify the user as well as to assist in the routing of the authentication request. Please note that the NAI may not necessarily be the same as the user's email address or the user identity submitted in an application layer authentication.

Network Access Server

The Network Access Server (NAS) is the device that clients connect to in order to get access to the network. In Point to Point Tunneling Protocol (PPTP) terminology, this is referred to as the PPTP Access Concentrator (PAC), and in Layer 2 Tunneling Protocol (L2TP) terminology, it is referred to as the L2TP Access Concentrator (LAC). In IEEE 802.11, it is referred to as an Access Point.

Home Network

The network which holds the authentication credentials for a user.

Visited Network

A network other than the home network, where the user attempts to gain network access. The Visited Network typically has a relationship with the Home Network, possibly through one or more intermediary proxies.

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Problem Statement

This section describes how RADIUS proxying works, how CoA packets work, and why CoA proxying as discussed in [RFC5176] is insufficient to create a working system.

2.1. Typical RADIUS Proxying

When a RADIUS server proxies an Access-Request packet, it typically does so based on the contents of the User-Name attribute, which contains a Network Access Identifier (NAI) [RFC7542]. This specification describes how to use the NAI in order to proxy CoA packets across multiple hops. Other methods of proxying CoA packets are possible, but are not discussed here.

In order to determine the "next hop" for a packet, the proxying server looks up the "Realm" portion of the NAI in a logical AAA routing table, as described in Section 3 of [RFC7542]. The entry in that table contains information about the "next hop" to which the packet is sent. This information can be IP address, shared secret, certificate, etc. The "next hop" may also be another proxy, or it may be the Home Server for that realm.

If the "next hop" is a proxy, that proxy will perform the same Realm lookup, and then proxy the packet as above. At some point, the "next hop" will be the Home Server for that realm.

The Home Server validates the NAI in the User-Name attribute against the list of Realms hosted by the Home Network. If there is no match, then an Access-Reject is returned. All other packets are processed through local site rules, which result in an appropriate response packet being sent. This response packet can be Access-Accept, Access-Challenge, or Access-Reject.

The RADIUS client receiving that response packet will match it to an outstanding request. If the client is part of a proxy, the proxy will then send that response packet in turn to the system that originated the Access-Request. This process occurs until the response packet arrives at the NAS.

The proxies are typically stateful with respect to ongoing request / response packets, but stateless with respect to user sessions. That is, once a response has been sent by the proxy, it can discard all information about the request packet, other than what is needed for detecting retransmissions as per Section 2.2.2 of [RFC5080].

The same method is used to proxy Accounting-Request packets. The combination of the two methods allows proxies to connect Visited

Networks to Home Networks for all AAA purposes.

2.2. CoA Processing

[RFC5176] describes how CoA clients send packets to CoA servers. We note that system comprising the CoA client is typically co-located with, or is the same as, the RADIUS server. Similarly, the CoA server is a system that is either co-located with, or is the same as, the RADIUS client.

In the case of packets sent inside of one network, the source and destination of CoA packets are locally determined. There is thus no need for standardization of that process, as networks are free to send CoA packets whenever they want, for whatever reason they want.

2.3. Failure of CoA Proxying

The situation is more complicated when proxies are involved. [RFC5176] suggests that CoA proxying is permitted, but that specification makes no suggestions for how that proxying should be done.

If proxies were to track user sessions, it would be possible for a proxy to match an incoming CoA packet to a user session, and then to proxy the CoA packet to the RADIUS client that originated the Access-Request for that session. There are many problems with such a scenario.

The CoA server may, in fact, not be co-located with the RADIUS client. In which case it may not have access to user session information for performing the reverse path forwarding.

The CoA server may be down, but there may be a different CoA server which could successfully process the packet. The CoA client should then fail over to a different CoA server. If the reverse path is restricted to be the same as the forward path, then such fail-over is not possible.

In a roaming consortium, the proxies may forward traffic for tens of millions of users. Tracking each user session can be expensive and complicated, and doing so does not scale well. For that reason, most proxies do not record user sessions.

Even if the proxy recorded user sessions, [RFC5176] is silent on the topic of what attributes constitute "session identification attributes". That silence means it is impossible for a proxy to determine if a CoA packet matches a particular user session.

The result of all of these issues is that CoA proxying is impossible when using the behavior defined in [RFC5176].

3. How to Perform CoA Proxying

The solution to the above problem is to use Realm-based proxying on the reverse path, just as with the forward path. In order for the reverse path proxying to work, the proxy decision must be based on an attribute other than User-Name.

The reverse path proxying can be done by using the Operator-Name attribute defined in [RFC5580], Section 4.1. We repeat a portion of that definition here for clarity:

This attribute carries the operator namespace identifier and the operator name. The operator name is combined with the namespace identifier to uniquely identify the owner of an access network.

Followed by a description of the REALM namespace:

REALM ('1' (0x31)):

The REALM operator namespace can be used to indicate operator names based on any registered domain name. Such names are required to be unique, and the rights to use a given realm name are obtained coincident with acquiring the rights to use a particular Fully Qualified Domain Name (FQDN). ...

In short, the Operator-Name attribute contains the an ASCII "1", followed by the Realm of the Visited Network. e.g. for the "example.com" realm, the Operator-Name attribute contains the text "1example.com". This information is precisely what is needed by intermediate nodes in order to perform CoA proxying.

The remainder of this document describes how CoA proxying can be performed by using the Operator-Name attribute. We describe how the forward path has to change in order to allow reverse path proxying. We then describe how reverse path proxying works. And we describe how Visited Networks and Home Networks have to behave in order for CoA proxying to work.

We note that as a proxied CoA packet is sent only to one destination, the Operator-Name attribute MUST NOT occur more than once in a packet. If a packet contains more than one Operator-Name, implementations MUST treat the second and subsequent attributes as "invalid attributes", as discussed in Section 2.8 of [RFC6929].

3.1. Changes to Access-Request and Accounting-Request packets

When a Visited Network proxies an Access-Request or Accounting-Request packet outside of its network, a Visited Network that wishes to support Realm-based CoA proxying SHOULD include an Operator-Name attribute in the packet, as discussed in Section 4.1 of [RFC5580]. The contents of the Operator-Name should be "1", followed by the realm name of the Visited Network. Where the Visited Network has more than one realm name, a "canonical" one SHOULD be chosen, and used for all packets.

Visited Networks MUST use a consistent value for Operator-Name for any one user session. That is, sending "1example.com" in an Access-Request packet, and "1example.org" in an Accounting-Request packet for that same session is forbidden. Such behavior would make it look like a single user session was active simultaneously in two different Visited Networks, which is impossible.

Proxies that record user session information SHOULD also record Operator-Name. Proxies that do not record user session information do not need to record Operator-Name.

Home Networks SHOULD record Operator-Name along with any other information that they record about user sessions. Home Networks that expect to send CoA packets to Visited Networks MUST record Operator-Name for each user session that originates from a Visited Network. Failure to record the Operator-Name would mean that the Home Network would not know where to send any CoA packet.

Networks that host both the RADIUS client and RADIUS server do not need to create, record or track Operator-Name. That is, if the Visited Network and Home Network are the same, there is no need to use the Operator-Name attribute.

3.2. Proxying of CoA-Request and Disconnect-Request packets

When a Home Network wishes to send a CoA-Request or Disconnect-Request packet to a Visited Network, it MUST include an Operator-Name attribute in the CoA packet. The value of the Operator-Name MUST be the value which was recorded earlier for that user session.

The Home Network MUST lookup the realm from the Operator-Name in a logical "realm routing table", as discussed in [RFC7542] Section 3. That logical realm table is defined there as:

a logical AAA routing table, where the "utf8-realm" portion acts as a key, and the values stored in the table are one or more "next hop" AAA servers.

In order to support proxying of CoA packets, this table is extended to include a mapping between "utf8-realm" and one or more "next hop" CoA servers.

When proxying CoA-Request and Disconnect-Request packets, the lookups will return data from the "CoA server" field, instead of the "AAA server" field.

In practice, this process means that CoA proxying works exactly like "normal" RADIUS proxying, except that the proxy decision is made using the realm from the Operator-Name attribute, instead of using the realm from the User-Name attribute.

Proxies that receive the CoA packet will look up the realm from the Operator-Name in a logical "realm routing table", as with Home Servers, above. The packet is then sent to the proxy for the realm which was found in that table. This process continues with any subsequent proxies until the packet reaches a public CoA server at the Visited Network.

Where the realm is unknown, the proxy MUST return a NAK packet that contains an Error-Cause attribute having value 502 ("Request Not Routable").

Proxies which receive a CoA packet MUST NOT use the NAI from the User-Name in order to make proxying decisions. Doing so would result in the CoA packet being forwarded to the Home Network, while the user's session is in the Visited Network.

We also update Section 5 of [RFC5580] to permit CoA-Request and Disconnect-Request packets to contain zero or one instances of the Operator-Name attribute.

3.3. Reception of CoA-Request and Disconnect-Request packets

After some proxying, the CoA packet will be recieved by the CoA server in the Visited Network. That CoA server MUST validate the NAI in the Operator-Name attribute against the list of realms hosted by the Visited Network. If the realm is not found, then the CoA server MUST return a NAK packet that contains an Error-Cause attribute having value 502 ("Request Not Routable").

Some Home Networks will not have permission to send CoA packets to the Visited Network. The CoA server SHOULD therefore also validate the NAI contained in the User-Name attribute. If the Home Network is not permitted to send CoA packets to this Visited Network, then the CoA server MUST return a NAK packet that contains an Error-Cause attribute having value 502 ("Request Not Routable").

These checks make it more difficult for a malicious Home Network to scan roaming network in order to determine which Visited Network hosts which Realm. That information should be known to all parties in advance, and exchanged via methods outside of this specification. Those methods will typically be in the form of contractual relationships between parties, or as membership in a roaming consortium.

The CoA server in the Visited Network will also ensure that the Operator-NAS-Identifier attribute is known, as described below. If the attribute matches a known NAS, then the packet will be sent to that NAS. Otherwise, the CoA server MUST return a NAK packet that contains an Error-Cause attribute having value 403 ("NAS Identification Mismatch").

All other received packets are processed as per local site rules, and will result in an appropriate response packet being sent. This process mirrors the method used to process Access-Request and Accounting-Request packets described above.

The processing by Visited Network will normally include sending the CoA packet to the NAS; having the NAS process it; and then returning any response packet back up the proxy chain to the Home Server.

The only missing piece here is the procedure by which the Visited Network gets the packet from its public CoA server to the NAS. The Visited Network could use NAS-Identifier, NAS-IP-Address, or NAS-IPv6-Address, but these attributes may have been edited by an intermediate proxy, or the attributes may be missing entirely.

These attributes may be incorrect because proxies forwarding Access-Request packets often re-write them for internal policy reasons. These attributes may be missing, because the Visited Network may not want all upstream proxies and Home Servers to have detailed information about the internals of its private network, and may remove them itself.

We therefore need a way to identify a NAS in the Visited Network, in a way which is both private, and which does not use any existing attribute. Our solution is to define an Operator-NAS-Identifier attribute, which identifies an individual NAS in the Visited Network.

3.4. Operator-NAS-Identifier

The Operator-NAS-Identifier attribute is an opaque token that identifies an individual NAS in a Visited Network. It MAY appear in the following packets: Access-Request, Accounting-Request, CoA-Request, or Disconnect-Request. Operator-NAS-Identifier MUST NOT

appear in any other packet.

Operator-NAS-Identifier MAY occur in a packet if the packet also contains an Operator-Name attribute. Operator-NAS-Identifier MUST NOT appear in a packet if there is no Operator-Name in the packet. As each proxied CoA packet is sent only to one NAS, the Operator-NAS-Identifier attribute MUST NOT occur more than once in a packet. If a packet contains more than one Operator-NAS-Identifier, implementations MUST treat the second and subsequent attributes as "invalid attributes", as discussed in Section 2.8 of [RFC6929].

An Operator-NAS-Identifer attribute SHOULD be added to an Access-Request or Accounting-Request packet by a Visited Network, before proxying a packet to an external RADIUS server. When the Operator-NAS-Identifer attribute is added to a packet, the following attributes SHOULD be deleted from the packet: NAS-IP-Address, NAS-IPv6-Address, NAS-Identifier. If these attributes are deleted, the proxy MUST then add a NAS-Identifier attribute, in order satisfy the requirements of Section 4.1 of [RFC2865], and Section 4.1 of [RFC2866]. The contents of the new NAS-Identifier SHOULD be the Realm name of the visited network.

When a server receives a packet that already contains an Operator-NAS-Identifer attribute, no such editing is performed.

The Operator-NAS-Attribute MUST NOT be added to any packet by any other proxy or server in the network. Only the Visited Network (i.e. the operator) can name a NAS which is inside of the Visited Network.

The result of these requirements is that for everyone outside of the Visited Network, there is only one NAS: the Visited Network itself. And, the Visited Network is able to identify its own NASes to its own satisfaction.

This usage of the Operator-NAS-Identifier attribute parallels the Operator-Name attribute which was defined in Section 4.1 of [RFC5580].

The Operator-NAS-Identifier attribute is defined as follows.

Description

An opaque token describing the NAS a user has logged into.

Type

TBD. To be assigned by IANA from the "short extended space".

Length

4 to 35.

Implementations supporting this attribute MUST be able to handle between one (1) and thirty-two (32) octets of data. Implementations creating an Operator-NAS-Identifier MUST NOT create attributes with more than sixty-four octets of data. A thirty-two octet string should be more than sufficient for future uses.

Data Type

string. See [RFC8044] Section 3.6 for a definition.

Value

The contents of this attribute are an opaque token interpretable only by the Visited Network.

This token MUST allow the Visited Network to direct the packet to the NAS for the user's session. In practice, this requirement means that the Visited Network has two practical methods to create the value.

The first method is to create an opaque token per NAS, and then to store that information in a database. The database can be configured to allow querying by NAS IP address in order to find the correct Operator-NAS-Identifier. The database can also be configured to allow querying by Operator-NAS-Identifier in order to find the correct NAS IP address.

The second method is to obfuscate the NAS IP address using information known locally by the Visited network; for example, by XORing it with a locally known secret key. The output of that obfuscation operation is data that can be used as the value of Operator-NAS-Identifier. On reception of a CoA packet, the locally-known information can be used to un-obfuscate the value of Operator-NAS-Identifier, in order to determine the actual NAS IP address.

Note that there is no requirement that the value of Operator-NAS-Identifier be checked for integrity. Modification of the value can only result in the erroneous transaction being rejected.

We note that the Access-Request and Accounting-Request packets often contain the MAC address of the NAS. There is therefore no requirement that Operator-NAS-Identifier obfuscate or hide in any

way the total number of NASes in a Visited Network. That information is already public knowledge.

4. Requirements

4.1. Requirements on Home Servers

The Operator-NAS-Identifier attribute MUST be stored by a Home Server along with any user session identification attributes. When sending a CoA packet for a user session, the Home Server MUST include verbatim any Operator-NAS-Identifier it has recorded for that session.

A Home Server MUST NOT send CoA packets for users of other networks. The next few sections describe how other participants in the RADIUS ecosystem can help to enforce this requirement.

4.2. Requirements on Visited Networks

A Visited Network which receives a CoA packet that will be proxied to a NAS MUST perform all of the operations required for proxies by Section 4.3.2. This requirement is because we assume that the Visited Network has a proxy in between the NAS and any external (i.e. third-party) proxy. Situations where a NAS sends packets directly to a third-party RADIUS server are outside of the scope of this specification.

The Visited Network uses the content of the Operator-NAS-Identifier attribute to determine which NAS will receive the packet.

The Visited Network MUST remove the Operator-Name and Operator-NAS-Identifier attributes from any CoA packet prior to sending that packet to the final CoA server (i.e. NAS). This step is necessary due to the limits of Section 2.3 of [RFC5176].

The Visited Network MUST also ensure that the CoA packet sent to the NAS contains one of the following attributes: NAS-IP-Address, NAS-IPv6-Address, or NAS-Identifier. This step is the inverse of the removal suggested above in Section 3.4.

In general, the NAS should only receive attributes which identify or modify a user's session. It is not appropriate to send a NAS attributes which are used only for inter-proxy signaling.

4.3. Requirements on Proxies

There are a number of requirements on proxies, both CoA proxies and RADIUS proxies. For the purpose of this section, we assume that each RADIUS proxy shares a common administration with a corresponding CoA proxy, and that the two systems can communicate electronically. There is no requirement for these systems to be co-located.

4.3.1. Security Requirements on Proxies

Section 6.1 of [RFC5176] has some security requirements on proxies that handle CoA-Request and Disconnect-Request packets:

... a proxy MAY perform a "reverse path forwarding" (RPF) check to verify that a Disconnect-Request or CoA-Request originates from an authorized Dynamic Authorization Client.

We strengthen that requirement by saying that a proxy MUST perform a "reverse path forwarding" (RPF) check to verify that a CoA packet originates from an authorized Dynamic Authorization Client. Without this check, a proxy may forward packets from misconfigured or malicious parties, and thus contribute to the problem instead of preventing it. Where the check fails, the proxy MUST return a NAK packet that contains an Error-Cause attribute having value 502 ("Request Not Routable").

Proxies that record user session information SHOULD verify the contents of a received CoA packet against the recorded data for that user session. If the proxy determines that the information in the packet does not match the recorded user session, it SHOULD return a NAK packet that contains an Error-Cause attribute having value 503 ("Session Context Not Found"). These checks cannot be mandated due to the fact that [RFC5176] offers no advice on which attributes are used to identify a user's session.

We recognize that because a RADIUS proxy will see Access-Request and Accounting-Request packets, it will have sufficient information to forge CoA packets. The RADIUS proxy will thus have the ability to subsequently disconnect any user who was authenticated through itself.

We suggest that the real-world effect of this security problem is minimal. RADIUS proxies can already return Access-Accept or Access-Reject for Access-Request packets, and can change authorization attributes contained in an Access-Accept. Allowing a proxy to change (or disconnect) a user session post-authentication is not substantially different from changing (or refusing to connect) a user

session during the initial process of authentication.

The largest problem is that there are no provisions in RADIUS for "end to end" security. That is, the Visited Network and Home Network cannot communicate privately in the presence of proxies. This limitation originates from the design of RADIUS for Access-Request and Accounting-Request packets. That limitation is then carried over to CoA-Request and Disconnect-Request packets.

We cannot therefore prevent proxies or Home Servers from forging CoA packets. We can only create scenarios where that forgery is hard to perform, and/or is likely to be detected, and/or has no effect.

4.3.2. Filtering Requirements on Proxies

Section 2.3 of [RFC5176] makes the following requirement for CoA servers:

In CoA-Request and Disconnect-Request packets, all attributes MUST be treated as mandatory.

These requirements are too stringent for a CoA proxy. Only the final CoA server (i.e NAS) can make a decision on which attributes are mandatory and which are not.

Instead, we say that for a CoA proxy, all attributes MUST NOT be treated as mandatory. Proxies implementing this specification MUST perform proxying based on Operator-Name. Other schemes are possible, but are not discussed here. Proxies SHOULD forward all packets as-is, with minimal changes.

We note that some NAS implementations currently treat signaling attributes as mandatory. For example, some NAS implementations will NAK any CoA packet that contains a Proxy-State attribute. While this behavior is based on a straightforward reading of the above text, it causes problems in practice.

We update Section 2.3 of [RFC5176] to say that in CoA-Request and Disconnect-Request packets, the NAS MUST NOT treat as mandatory any attribute which is known to not affect the users session. For example, the Proxy-State attribute. Proxy-State is an attribute used for proxy-to-proxy signaling. It cannot affect the user's session, and therefore Proxy-State (and similar attributes) MUST be ignored by the NAS.

When Operator-Name and/or Operator-NAS-Identifier are received by a proxy, the proxy MUST pass those attributes through unchanged. This requirement applies to all proxies, including ones that forward any

or all of Access-Request, Accounting-Request, CoA-Request, and Disconnect-Request packets.

All attributes added by a RADIUS proxy when sending packets from the Visited Network to the Home Network Network MUST be removed by the corresponding CoA proxy from packets traversing the reverse path. That is, any attribute editing that is done on the "forward" path MUST be undone on the "reverse" path.

The result is that a NAS will only ever receive CoA packets that either contain attributes sent by the NAS to it's local RADIUS server, or contain attributes that are sent by the Home Server in order to perform a change of authorization.

Finally, we extend the above requirement not only to Operator-Name and Operator-NAS-Identifier, but also to any future attributes that are added for proxy-to-proxy signaling.

5. Functionality

This section describes how the two attributes work together to permit CoA proxying.

5.1. User Login

In this scenario, we follow a roaming user who is attempting to log in to a Visited Network. The login attempt is done via a NAS in the Visited Network. That NAS will send an Access-Request packet to the visited RADIUS server. The visited RADIUS server will see that the user is roaming, and will add an Operator-Name attribute, with value "1" followed by it's own realm name. e.g. "1example.com". The visited RADIUS server MAY also add an Operator-NAS-Identifier. The NAS identification attributes are also edited, as required by Section 3.4, above.

The Visited Server will then proxy the authentication request to an upstream server. That server may be the Home Server, or it may be a proxy. In the case of a proxy, the proxy will forward the packet, until the packet reaches the Home Server.

The Home Server will record the Operator-Name and Operator-NAS-Identifier along with other information about the users session, if those attributes are present in a packet.

5.2. CoA Proxying

At some later point in time, the Home Server determines that a user session should have its authorization changed, or be disconnected. The Home Server looks up the Operator-Name and Operator-NAS-

Identifier, along with other user session identifiers as described in [RFC5176]. The Home Server then looks up the realm from the Operator-Name attribute in the logical AAA routing table, in order to find the "next hop" CoA server for that realm (that may be a proxy). The CoA request is then sent to that CoA server.

The CoA server receives the request, and if it is a proxy, performs a similar lookup as done by the Home Server. The packet is then proxied repeatedly until it reaches the Visited Network.

If the proxy cannot find a destination for the request, or if no Operator-Name attribute exists in the request, the proxy will return a CoA-NAK with Error-Cause 502 (Request Not Routable).

The Visited Network will receive the CoA-Request packet, and will use the Operator-NAS-Identifier (if available) attribute to determine which local CoA server (i.e. NAS) the packet should be sent to. If there is no Operator-NAS-Identifier attribute, the Visited Network may use other means to locate the NAS, such as consulting a local database which tracks user sessions.

The Operator-Name and Operator-NAS-Identifier attributes are then removed from the packet; one of NAS-IP-Address, or NAS-IPv6-Address, or NAS-Identifier is added to the packet; and the packet is then sent to the CoA server.

If no CoA server can be found, the Visited Network return a CoA-NAK with Error-Cause 403 (NAS Identification Mismatch).

Any response from the CoA server (NAS) is returned to the Home Network, via the normal method of returning responses to requests.

6. Security Considerations

This specification incorporates by reference the Section 11 of [RFC6929]. In short, RADIUS has many known issues which are discussed in detail there, and which do not need to be repeated here.

This specification adds one new attribute, and defines new behavior for RADIUS proxying. As this behavior mirrors existing RADIUS proxying, we do not believe that it introduces any new security issues. We note, however, that RADIUS proxying has a series of inherent security issues.

6.1. RADIUS Security and Proxies

The requirement that packets be signed with a shared secret means that a CoA packet can only be received from a trusted party. Or transitively, received from a third party via a trusted party. This security provision of the base RADIUS protocol makes it impossible for untrusted parties to affect the user's session.

When RADIUS proxying is performed, all packets are signed on a hop-by-hop basis. Any intermediate proxy can therefore forge packets, replay packets, or modify the contents of any packets entirely without detection. As a result, the secure operation of such a system depends largely on trust, instead of on technical means.

CoA packet proxying has all of the same issues as noted above. We note that the proxies which see and can modify CoA packets are generally the same proxies which can see or modify Access-Request and Accounting-Request packets. As such, there are few additional security implications in allowing CoA proxying.

The main security implication left is that Home Networks now have the capability to disconnect, or change the authorization of users in a Visited Network. As this capability is only enabled when mutual agreement is in place, and only for those parties who can already control the users's session, there are no new security issues with this specification.

6.2. Security of the Operator-NAS-Identifier Attribute

Nothing in this specification depends on the security of the Operator-NAS-Identifier attribute. The entire process would work exactly the same if the Operator-NAS-Identifier simply contained the NAS IP address that is hosting the user's session. The only real downside in that situation would be that external parties would see some additional private information about the Visited Network. They would still, however, be unable to leverage that information to do anything malicious.

The main reason to use an opaque token for the Operator-NAS-Identifier is that there is no compelling reason to make the information public. We therefore recommend that the value be simply an opaque token. We also state that there is no requirement for integrity protection or replay detection of this attribute. The rest of the RADIUS protocol ensures that modification or replay of the Operator-NAS-Identifier will either have no effect, or will have the same effect as if the value had not been modified.

Trusted parties can modify a user's session on the NAS only when they

have sufficient information to identify that session. In practice, this limitation means that those parties already have access to the users's session information. Which is to say, those parties are the proxies who are already forwarding Access-Request and Accounting-Request packets.

Since those parties already have the ability to see and modify all of the information about a user's session, there is no additional security issue with allowing them to see and modify CoA packets.

In short, any security issues with the contents of Operator-NAS-Identifier are largely limited by the security of the underlying RADIUS protocol. This limitation means that it does not matter how the values of Operator-NAS-Identifier are created, stored, or used.

7. IANA Considerations

IANA is instructed to allocate one new RADIUS attribute, as per Section 3.3, above. The Operator-NAS-Identifier attribute is to be allocated from the RADIUS Attribute Types registry as follows:

Value: [TBD-at-Registration]
Description: Operator-NAS-Identifier
Data Type: string
Reference: [RFC-to-be]

8. References

8.1. Normative References

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", 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>>.

[RFC5080]

Nelson, D., and DeKok, A., "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. et al, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January

2008, <<http://www.rfc-editor.org/info/rfc5176>>.

[RFC5580]

Tschofenig H., Ed. "Carrying Location Objects in RADIUS and Diameter", RFC 5580, August 2009, <<http://www.rfc-editor.org/info/rfc5580>>.

[RFC6929]

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

[RFC7542]

DeKok A., "The Network Access Identifier", RFC 7542, May 2015, <<http://www.rfc-editor.org/info/rfc7542>>.

[RFC8044]

DeKok A., "Data Types in the Remote Authentication Dial-In User Service Protocol (RADIUS)", RFC 8044, January 2017, <<http://www.rfc-editor.org/info/rfc8044>>.

[RFC8174]

Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174, May 2017, <<http://www.rfc-editor.org/info/rfc8174>>.

8.2. Informative References

[RFC2866]

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

Authors' Addresses

Alan DeKok
The FreeRADIUS Server Project

Email: aland@freeradius.org

Jouni Korhonen

EMail: jouni.nospam@gmail.com

