Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 22, 2016

M. Boucadair
C. Jacquenet
Orange

January 19, 2016

RADIUS Extensions for Network-Assisted Multipath TCP (MPTCP)
draft-boucadair-mptcp-radius-01

Abstract

One of the promising deployment scenarios for Multipath TCP (MPTCP) is to enable a Customer Premises Equipment (CPE) that is connected to multiple networks (e.g., DSL, LTE, WLAN) to optimize the usage of its network attachments. Because of the lack of MPTCP support at the server side, some service providers consider a network-assisted model that relies upon the activation of a dedicated function called: MPTCP Concentrator.

This document specifies a new Remote Authentication Dial-In User Service (RADIUS) attributes that carry the IP addresses that allow CPE devices to reach one or multiple MPTCP Concentrators.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on July 22, 2016.

Table of Contents

1. Introduction

One of the promising deployment scenarios for Multipath TCP (MPTCP, [RFC6824]) is to enable a Customer Premises Equipment (CPE) that is connected to multiple networks (e.g., DSL, LTE, WLAN) to optimize the usage of such resources, see for example [RFC4908]. This deployment scenario relies on MPTCP proxies located on both the CPE and network sides (Figure 1). MPTCP Proxies deployed in the network play the role of traffic concentrator.

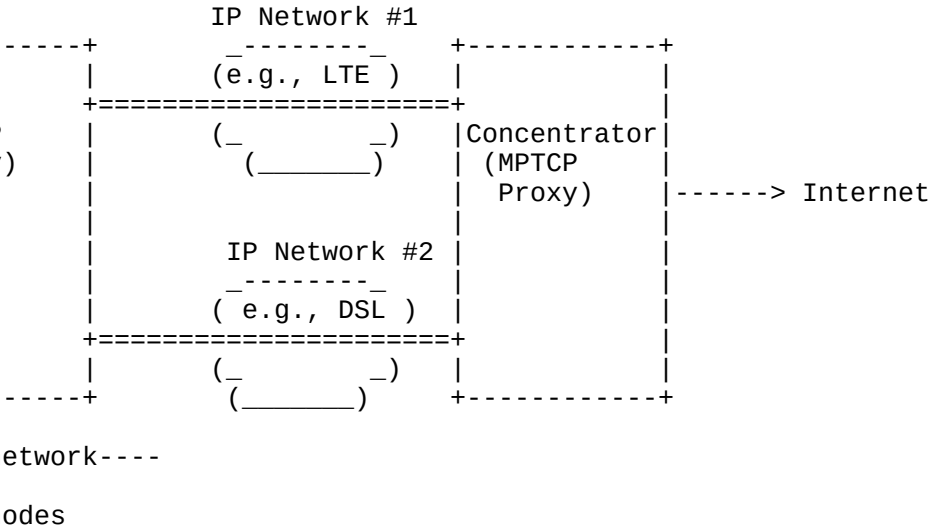


Figure 1: "Network-Assisted" MPTCP Design

Within this document, an MPTCP Concentrator (or concentrator) refers to a functional element that is responsible for aggregating the traffic originated by a group of CPEs. This element is located in the network. One or multiple concentrators can be deployed in the network to assist MPTCP-enabled CPEs to establish MPTCP connections via their available network attachments. On the uplink path, the concentrator terminates the MPTCP connections [RFC6824] received from its customer-facing interfaces and transforms these connections into legacy TCP connections [RFC0793] towards upstream servers. On the downlink path, the concentrator turns the legacy server's TCP connection into MPTCP connections towards its customer-facing interfaces.

Both implicit (where a CPE has no specific knowledge of any concentrator deployed in the network) and explicit modes are considered to steer traffic towards an MPTCP Concentrator. This document focuses on the explicit mode that consists in explicitly configuring a CPE with the reachability information of a MPTCP concentrator.

This document specifies two new Remote Authentication Dial-In User Service (RADIUS, [RFC2865]) attributes that carry the MPTCP Concentrator IP address list (Section 2). In order to accommodate both IPv4 and IPv6 deployment contexts, and given the constraints in Section 3.4 of [RFC6158], two attributes are specified. Note that one or multiple IPv4 and/or IPv6 addresses may be returned to a requesting CPE. A sample use case is described in Section 3.

This document assumes that the MPTCP concentrator(s) reachability information can be stored in Authentication, Authorization, and Accounting (AAA) servers while the CPE configuration is usually provided by means of DHCP ([RFC2131][RFC3315]).

This specification assumes an MPTCP Concentrator is reachable through one or multiple IP addresses. As such, a list of IP addresses can be communicated via RADIUS. Also, it assumes the various network attachments provided to an MPTCP-enabled CPE are managed by the same administrative entity.

This document adheres to [I-D.ietf-radext-datatypes] for defining the new attributes.

2. MPTCP RADIUS Attributes

2.1. MPTCP-IPv4-Concentrator

Description

The RADIUS MPTCP-Concentrator-IPv4 attribute contains the IPv4 address of an MPTCP Concentrator that is assigned to a CPE.

Because multiple MPTCP Concentrator IP addresses may be provisioned to an authorised CPE (that is a CPE entitled to solicit the resources of a concentrator to establish MPTCP connections), multiple instances of the MPTCP-Concentrator-IPv4 attribute MAY be included; each instance of the attribute carries a distinct IP address.

Both MPTCP-Concentrator-IPv4 and MPTCP-Concentrator-IPv6 attributes MAY be present in a RADIUS message.

The MPTCP-Concentrator-IPv4 Attribute MAY appear in a RADIUS Access-Accept packet. It MAY also appear in a RADIUS Access-Request packet as a hint to the RADIUS server to indicate a preference, although the server is not required to honor such a hint.

The MPTCP-Concentrator-IPv4 Attribute MAY appear in a CoA-Request packet.

The MPTCP-Concentrator-IPv4 Attribute MAY appear in a RADIUS Accounting-Request packet.

The MPTCP-Concentrator-IPv4 Attribute MUST NOT appear in any other RADIUS packet.

Type

TBA (see Section 6).

Length

6

Data Type

The attribute MPTCP-Concentrator-IPv4 is of type ip4addr (Section 3.3 of [I-D.ietf-radext-datatypes]).

Value

This field includes an IPv4 address (32 bits) of the MPTCP Concentrator.

The MPTCP-Concentrator-IPv4 attribute MUST NOT include multicast and host loopback addresses [RFC6890]. Anycast addresses are allowed to be included in an MPTCP-Concentrator-IPv4 attribute.

2.2. MPTCP-IPv6-Concentrator

Description

The RADIUS MPTCP-Concentrator-IPv6 attribute contains the IPv6 address of an MPTCP Concentrator that is assigned to a CPE.

Because multiple MPTCP Concentrator IP addresses may be provisioned to an authorised CPE (that is a CPE entitled to solicit the resources of a concentrator to establish MPTCP connections), multiple instances of the MPTCP-Concentrator-IPv6 attribute MAY be included; each instance of the attribute carries a distinct IP address.

Both MPTCP-Concentrator-IPv4 and MPTCP-Concentrator-IPv6 attributes MAY be present in a RADIUS message.

The MPTCP-Concentrator-IPv6 Attribute MAY appear in a RADIUS Access-Accept packet. It MAY also appear in a RADIUS Access-Request packet as a hint to the RADIUS server to indicate a preference, although the server is not required to honor such a hint.

The MPTCP-Concentrator-IPv6 Attribute MAY appear in a CoA-Request packet.

The MPTCP-Concentrator-IPv6 Attribute MAY appear in a RADIUS Accounting-Request packet.

The MPTCP-Concentrator-IPv6 Attribute MUST NOT appear in any other RADIUS packet.

Type

TBA (see Section 6).

Length

18

Data Type

The attribute MPTCP-Concentrator-IPv6 is of type ip6addr (Section 3.9 of [I-D.ietf-radext-datatypes]).

Value

This field includes an IPv6 address (128 bits) of the MPTCP concentrator.

The MPTCP-Concentrator-IPv6 attribute MUST NOT include multicast and host loopback addresses [RFC6890]. Anycast addresses are allowed to be included in an MPTCP-Concentrator-IPv6 attribute.

3. Sample Use Case

This section does not aim to provide an exhaustive list of deployment scenarios where the use of the RADIUS MPTCP-Concentrator-IPv6 and MPTCP-Concentrator-IPv4 attributes can be helpful. Typical deployment scenarios are described, for instance, in [RFC6911].

Figure 2 shows an example where a CPE is assigned an MPTCP Concentrator. This example assumes that the Network Access Server (NAS) embeds both RADIUS client and DHCPv6 server capabilities.

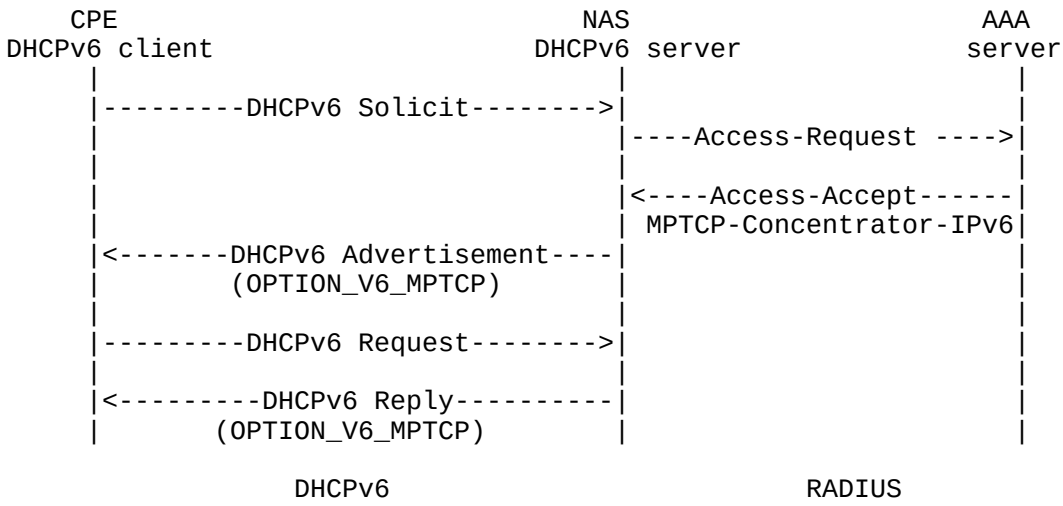


Figure 2: Sample Flow Example (1)

Upon receipt of the DHCPv6 Solicit message from a CPE, the NAS sends a RADIUS Access-Request message to the AAA server. Once the AAA server receives the request, it replies with an Access-Accept message (possibly after having sent a RADIUS Access-Challenge message and assuming the CPE is entitled to connect to the network) that carries a list of parameters to be used for this session, and which include MPTCP Concentrator reachability information (namely a list of IP addresses).

The content of the MPTCP-Concentrator-IPv6 attribute is then used by the NAS to complete the DHCPv6 procedure that the CPE initiated to retrieve information about the MPTCP Concentrator it has been assigned.

Upon change of the MPTCP Concentrator assigned to a CPE, the RADIUS server sends a RADIUS CoA message [RFC5176] that carries the RADIUS MPTCP-Concentrator-IPv6 attribute to the NAS. Once that message is accepted by the NAS, it replies with a RADIUS CoA ACK message. The NAS replaces the old MPTCP Concentrator with the new one.

Figure 3 shows another example where a CPE is assigned an MPTCP Concentrator, but the CPE uses DHCPv6 to retrieve a list of IP addresses of an MPTCP concentrator.

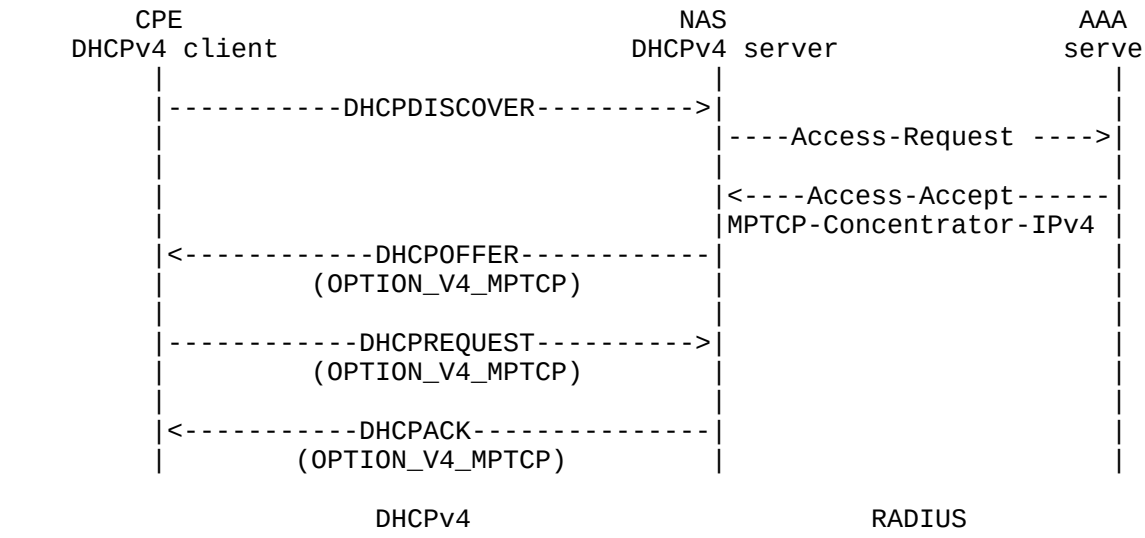


Figure 3: Sample Flow Example (2)

Some deployments may rely on the mechanisms defined in [RFC4014] or [RFC7037], which allows a NAS to pass attributes obtained from a RADIUS server to a DHCP server.

4. Security Considerations

RADIUS-related security considerations are discussed in [RFC2865].

MPTCP-related security considerations are discussed in [RFC6824] and [RFC6181].

Traffic theft is a risk if an illegitimate concentrator is inserted in the path. Indeed, inserting an illegitimate concentrator in the forwarding path allows to intercept traffic and can therefore provide access to sensitive data issued by or destined to a host. To mitigate this threat, secure means to discover a concentrator should be enabled.

5. Table of Attributes

The following table provides a guide as what type of RADIUS packets that may contain these attributes, and in what quantity.

Access-Request	Access-Accept	Access-Reject	Challenge	Acct. # Request	Attribute
0+	0+	0	0	0+	TBA MPTCP-Concentrator-IPv4
0+	0+	0	0	0+	TBA MPTCP-Concentrator-IPv6

CoA-Request	CoA-ACK	CoA-NACK	#	Attribute
0+	0	0		TBA MPTCP-Concentrator-IPv4
0+	0	0		TBA MPTCP-Concentrator-IPv6

The following table defines the meaning of the above table entries:

0 This attribute MUST NOT be present in packet.
0+ Zero or more instances of this attribute MAY be present in packet.

6. IANA Considerations

IANA is requested to assign two new RADIUS attribute types from the IANA registry "Radius Attribute Types" located at http://www.iana.org/assignments/radius-types:

MPTCP-Concentrator-IPv4 (TBA)

MPTCP-Concentrator-IPv6 (TBA)

7. Acknowledgements

Thanks to Alan DeKok for the comments.

8. References

8.1. Normative References

[I-D.ietf-radext-datatypes] DeKok, A., "Data Types in the Remote Authentication Dial-In User Service Protocol (RADIUS)", draft-ietf-radext-datatypes-02 (work in progress), November 2015.

[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, DOI 10.17487/RFC2865, June 2000, <http://www.rfc-editor.org/info/rfc2865>.

[RFC6158] DeKok, A., Ed. and G. Weber, "RADIUS Design Guidelines", BCP 158, RFC 6158, DOI 10.17487/RFC6158, March 2011, <http://www.rfc-editor.org/info/rfc6158>.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <http://www.rfc-editor.org/info/rfc6890>.

8.2. Informative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <http://www.rfc-editor.org/info/rfc2131>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC4014] Droms, R. and J. Schnizlein, "Remote Authentication Dial-In User Service (RADIUS) Attributes Suboption for the Dynamic Host Configuration Protocol (DHCP) Relay Agent Information Option", RFC 4014, DOI 10.17487/RFC4014, February 2005, <http://www.rfc-editor.org/info/rfc4014>.

[RFC4908] Nagami, K., Uda, S., Ogashiwa, N., Esaki, H., Wakikawa, R., and H. Ohnishi, "Multi-homing for small scale fixed network Using Mobile IP and NEMO", RFC 4908, DOI 10.17487/RFC4908, June 2007, <http://www.rfc-editor.org/info/rfc4908>.

[RFC5176] Chiba, M., Dommety, G., Eklund, M., Mitton, D., and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, DOI 10.17487/RFC5176, January 2008, <http://www.rfc-editor.org/info/rfc5176>.

[RFC6181] Bagnulo, M., "Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6181, DOI 10.17487/RFC6181, March 2011, <http://www.rfc-editor.org/info/rfc6181>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <http://www.rfc-editor.org/info/rfc6824>.

[RFC6911] Dec, W., Ed., Sarikaya, B., Zorn, G., Ed., Miles, D., and B. Lourdelet, "RADIUS Attributes for IPv6 Access Networks", RFC 6911, DOI 10.17487/RFC6911, April 2013, <http://www.rfc-editor.org/info/rfc6911>.

[RFC7037] Yeh, L. and M. Boucadair, "RADIUS Option for the DHCPv6 Relay Agent", RFC 7037, DOI 10.17487/RFC7037, October 2013, <http://www.rfc-editor.org/info/rfc7037>.

Authors' Addresses

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Christian Jacquenet
Orange
Rennes
France

Email: christian.jacquenet@orange.com

Network Working Group
Internet Draft
Intended Status: Standards Track
Expiration Date: September 28, 2017

E. Chen
N. Shen
Cisco Systems

March 27, 2017

RADIUS Identifier Attribute
draft-chen-radext-extended-header-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/1id-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 September 28, 2017.

Copyright Notice

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 ("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 identifier parameter in subsequent RADIUS packets.

The attribute also provides an option for carrying the RADIUS packet type "Code" in a larger field just in case that becomes necessary in the future.

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

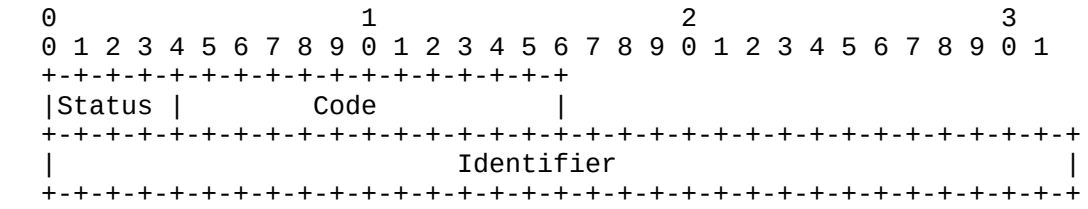
1.1. Specification of Requirements

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].

2. Protocol Extensions

2.1. The Identifier Attribute

A new attribute, termed "Identifier Attribute", is specified which can be used to discover the support for the Extended Identifier feature between a client and a server. It can also be used to carry the Identifier field (and optionally the Code field) in a RADIUS packet after the support is confirmed. The attribute number is TBD. The value of the attribute has 6 octets, and consists of the following fields:



where the 4-bit Status field is to be used in a Status-Server message to discover the support for the Extended Identifier feature. The following settings are defined:

- It is set to 1 (for "request") when a client sends the Status-Server request to a server indicating its support for the Extended Identifier feature.

- It is set to 2 (for "accept") or 3 (for "reject") by the server in its response to indicate whether it supports the Extended Identifier feature.

The 12-bit Code field, when needed, can be used in lieu of the Code field in the RADIUS packet header. The field is unused if its value is zero. The field is in use otherwise.

The Identifier field is a 4-octet unsigned integer. It is to be used in lieu of the Identifier field in the RADIUS packet header.

When the "Identifier Attribute" is used in a Status-Server request or reply, only the Status field is used. All other fields SHOULD be set to zero by the sender and MUST be ignored by the receiver.

When the "Identifier Attribute" is used in a message other than the Status-Server request or reply, the Status field is unused, and SHOULD be set to zero by the sender and MUST be ignored by the receiver.

Other than the larger sizes for the Identifier field and optionally for the Code field, these two fields remain unchanged semantically as defined in RFC 2865 [RFC2865] (and subsequent documents using the same packet format).

To simplify packet processing and for consistency, the "Identifier Attribute" MUST be encoded as the very first attribute in 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 "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 "Identifier Attribute", a client implementing this specification SHOULD first send a Status-Server request with the "Identifier Attribute" to indicate its support for the Extended Identifier feature.

When a server implementing this specification receives a Status-Server request with the "Identifier Attribute", it MUST include the "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 "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 "Identifier

Attribute" to a server until it has received a response from the server confirming its support for the Extended Identifier feature using the "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. Co-existence of Identifier Fields

After the functionality defined in this specification is discovered between the client and the server, RADIUS packets can be exchanged using either the Identifier field in the RADIUS packet header (without the "Identifier Attribute" in the packet), or the Identifier field in the "Identifier Attribute" as the very first attribute in the attribute list.

When the "Identifier Attribute" is present in a RADIUS packet other than the Status-Server request or reply, the Identifier field in the attribute MUST be used in lieu of the Identifier field in the RADIUS packet header. Similarly the Code field in the attribute, if it is non-zero, MUST be used in lieu of the Code field in the RADIUS packet header.

When the "Identifier Attribute" is used to carry the Identifier field, for better debugging it is RECOMMENDED that 255 be used in the Identifier field of the RADIUS packet header. Similarly it is RECOMMENDED that 255 be used in the Code field of the RADIUS packet header when the attribute is used to carry the Code field as well.

To simplify implementation, it is RECOMMENDED that the numbers 256 and larger be used as the "Identifier" in the "Identifier Attribute".

In response to a request from a client, the server SHOULD format the Identifier field in the same way as in the request, i.e., using either the Identifier field in the RADIUS packet header or the one in the "Identifier Attribute".

3. IANA Considerations

A new attribute ("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

TBD

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
Intended status: Experimental
Expires: November 3, 2017

D. Garcia
R. Marin
University of Murcia
A. Kandasamy
A. Pelov
Acklio

May 2, 2017

LoRaWAN Authentication in RADIUS
draft-garcia-radext-radius-lorawan-03

Abstract

This document describes a proposal for adding LoRaWAN support in RADIUS. The purpose is to integrate the LoRaWAN network join procedure with an Authentication, Authorization and Accounting (AAA) infrastructure based on RADIUS.

Status of This Memo

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

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

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

This Internet-Draft will expire on November 3, 2017.

Table of Contents

1. Introduction

Low Power Wide Area Network (LP-WAN) groups several radio technologies that allow communications with nodes far from the central communication endpoint (base station) in the range of kilometers depending on the specifics of the technology and the scenario. They are fairly recent and the protocols to manage those infrastructures are in continuous development. In some cases they may not consider aspects such as key management or directly tackle scalability issue in terms of authentication and authorization. The nodes to be authenticated and authorized is expected to be considerably high in number. One of the protocols that provide a complete solution is LoRaWAN [LoRaWAN]. LoRaWAN is a MAC layer protocol that use LoRa as its physical medium to cover long range (up-to 20 km depending on the environment) devices. LoRaWAN is designed for large scale networks and currently has a central entity

called Network Server which maintains a pre-configured key named AppKey for each of the devices on the network. Furthermore, session keys such as NwkSKey and AppSKey used for encryption of data messages, are derived with the help of this AppKey. Since each service provider would operate their Network Server individually, authenticating the devices becomes a tedious process because of inter-interoperability or the roaming challenges between the operators. An illustration of the LoRaWAN architecture can be seen in figure Figure 1. As we know the AAA infrastructure provides a flexible, scalable solution. They offer an opportunity to manage all these processes in a centralized manner as happens in other type of networks (e.g. cellular, WiFi, etc...) making it an interesting asset when integrated into the LoRaWAN architecture.

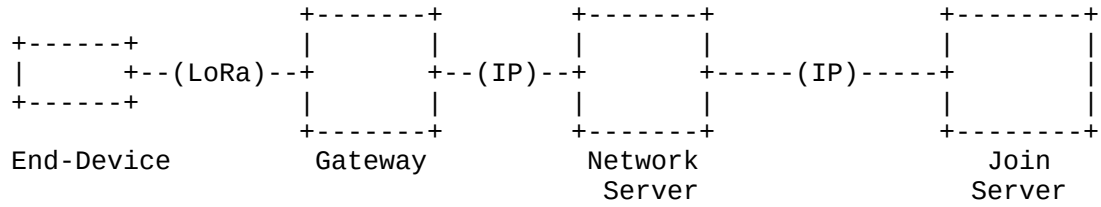


Figure 1: LoRAWAN Architecture

The End-Device communicates with the Gateway by using the LoRa modulation. The Gateway acts as a simple transceiver, which forwards all data do the Network Server, which performs the processing of the frames, network frame authentication (MIC verification), and which serves as Network Access Port. This document describes a way to use standard RADIUS servers as a Join Server, and to use the RADIUS protocol for the interaction between the Network Server and the Application Server. This integration is illustrated in figure Figure 2

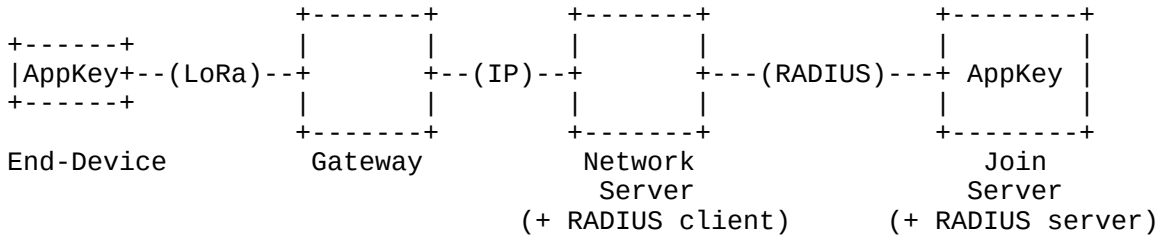


Figure 2: LoRAWAN Architecture with AAA and RADIUS authentication. End-Device and RADIUS server have a shared secret - the AppKey, which is used to derive the session keys (NwkSKey and AppSKey).

The document describes how LoRaWAN join procedure is integrated with AAA infrastructure using RADIUS [RFC2865] by defining the new attributes needed to support the LoRaWAN exchange.

1.1. Requirements Language

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

2. LoRaWAN support in RADIUS

Regarding the overall functionality, the RADIUS LoRaWAN support defines the new Attributes needed for the management of the join procedure. The Network Server will implement a RADIUS client supporting this specification and therefore, it MUST implement the RADIUS attributes for this service. The NAS-Port-Type specifying the type of port on which the Network Server is authenticating the End-Device in this case MAY be 18 (Wireless - Other) or a new one specifically assigned for LoRaWAN (TBD.).

3. LoRaWAN Overview

3.1. Introduction

The LoRAWAN specification defines how the MAC and PHY layer will be used with the LoRa radio technologies. It defines a process by which the smart objects can securely join the network in an authenticated way and exchange application information ciphered and integrity protected. The focus of this document is to extend how the process of joining is performed by the specification including a AAA infrastructure (RADIUS) to accomplish this. Next we review how the keys, and each message is used in the joining procedure. Then we elaborate some assumptions to design the integration of AAA in the joining procedure possible.

3.2. LoRaWAN join procedure Key Material

The LoRaWAN specification describes 3 keys involved in the joining procedure. One as a root key that will be used to generate the other two, which will be used to secure the message exchanges after the joining procedure success. The AppKey key used to derive the other two keys, NwkSKey and AppSKey:

- The AppKey is an AES-128 application specific key assigned by the owner of the application. This key is derived from an application-specific root key that is only known to the

application owner and is stored in each device and in the Join Server that will perform the authentication.

- The NwkSKey is a network session key that is specific to each End-Device. It is shared between the Network Server and the End-Device and used to calculate and verify the Message Integrity Code (MIC) for each data message, between both entities. Furthermore, it is used to cipher and decipher the payload of MAC-only data message.

- The AppSKey is an application session key specific to each End-Device. It is in charge of ciphering and deciphering the payload of application-specific data messages and is also used to calculate and verify the MIC that may be added to the payload of application-specific data messages.

3.3. LoRaWAN joining procedure

The LoRaWAN joining procedure, as described in the LoRaWAN Specification 1.0 [LoRaWAN], consists on one exchange. The first message of this exchange is called join-request (JR) message and is sent from the End-Device to the Network Server containing the AppEUI and DevEUI of the End-Device with a nonce of 2 octets called DevNonce. Figure 3 summarizes the format.

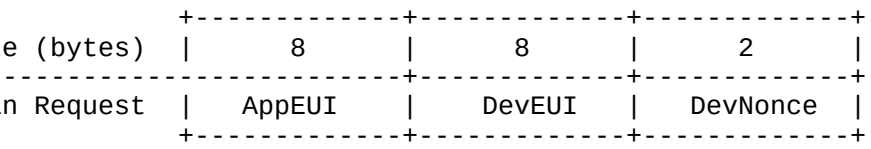


Figure 3: Join Request Message

In response to the join-request, the other endpoint will answer with the join-accept (JA) (Figure 4) if the End-Device is successfully authenticated and authorized to join the network. The join-accept contains a nonce (AppNonce), a network identifier (NetID), an End-Device address (DevAddr), a delay between the TX and RX (RxDelay) and, optionally, the CFList (see LoRaWAN specification [LoRaWAN] section 7).

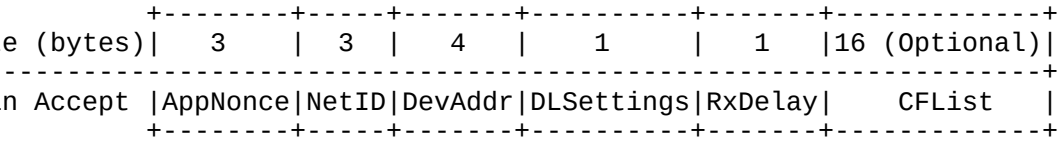


Figure 4: Join Accept Message

Next, we enumerate and describe each field involved in the join procedure message exchange.

- AppEUI: Global application ID in IEEE EUI64 to uniquely identify the application provider.
- DevEUI: Global End-Device ID in IEEE EUI64 to uniquely identify the End-Device
- DevNonce: A random value.
- AppNonce: A random value or some kind of unique ID provided by the Network Server. This value can be also generated by the AAA server, in case the network server wants to rely on the AAA server pseudo random number generation. For this, the AppNonce would be empty (set to zero), signaling the AAA server it has to generate the AppNonce.
- NetID: A network identifier
- DevAddr: A 32 bit identifier of the End-Device in the current network. It is composed of the Network ID and the Network Address.
- DLSettings: 8 bits containing the down-link configuration.
- RxDelay: 8 bits containing the delay between TX and RX.
- CFList (Optional): Channel frequency list.

3.4. LoRaWAN Key Derivation

The keys NwkSKey and AppSKey are derived from the AppKey in both the Join Server and the End-Device according to the LoRaWAN specification [LoRaWAN] as follows:

Derivation of the NwkSkey:

NwkSKey = aes128_encrypt(AppKey, 0x01 | AppNonce | NetID | DevNonce | pad16)

Derivation of the AppSkey:

AppSKey = aes128_encrypt(AppKey, 0x02 | AppNonce | NetID | DevNonce | pad16)

Note: The pad16 function appends octets of containing "zero" so that the length of the data is a multiple of 16.

4. Integration Overview

4.1. Mapping LoRaWAN Entities to AAA Infrastructure

In the current specification of LoRaWAN [LoRaWAN], there is no explicit reference to an external entity to which the Network Server can go to authenticate the End-Device. However, ongoing work related to LoRaWAN, such as the work in the LoRa Alliance [LoRaAllianceSecurity] sketches the use of a new entity, the Join Server, that will be in charge of performing the authentication. This separation of responsibilities is also the aim of our work, where the Join Server acts as an external AAA server in a AAA infrastructure using RADIUS as the protocol to communicate the Network Server and the Join Server. Further, it is under consideration the distribution of the AppSKey to a target application server instead of the Network Server. Therefore, the Join Server would need another protocol to deliver the AppSKey. Another RADIUS interface could be used for this purpose, though this I-D focuses on the joining procedure so far.

4.2. Assumptions

For the integration of LoRaWAN joining procedure with RADIUS next we describe some assumptions regarding the LoRaWAN specification. The first is that the AppKey is only shared between the AAA server (Join Server) and the End-Device. The outcome of the successful join procedure (i.e. NwkSKey and AppSKey) are sent from the AAA server to the network-server. This allows for the End-Device to exchange message with the network-server, once the join procedure is finished, as specified in LoRaWAN [LoRaWAN].

4.3. Protocol Exchange

The join procedure between the End-Device and the network-server entails one exchange consisting on a join-request message and a join-response message. In RADIUS the network-server implements a RADIUS client to communicate with the Join Server, which act as AAA Server.

The protocol exchange is done in the following steps:

1. The End-Device sends the join-request message to to the Network Server.

2. Upon reception of the LoRaWAN join-request message, the Network Server creates a RADIUS Access-Request message, with the Join-Request attribute containing the original message from the End-Device, and the Join-Answer Attribute with all the fields of a join-answer message except for the MIC, which will be calculated

by the AAA Server (Join Server), since is the one that holds the AppKey.

3. Once the AAA Server authenticates and authorizes the End-Device, sends back the Join-Answer with the MIC generated as specified by the LoRaWAN specification. Furthermore, as a consequence of a successful join procedure, the AppSKey (optional) and NwkSKey are generated and sent along in AppSKey and NwkSKey Attributes respectively.

4. The Network Server receives the Access-Accept (if successful), obtains the content of the Join-Request attribute and sends it to the End-Device, storing in association with that End-Device the NwkSKey and the AppSKey.

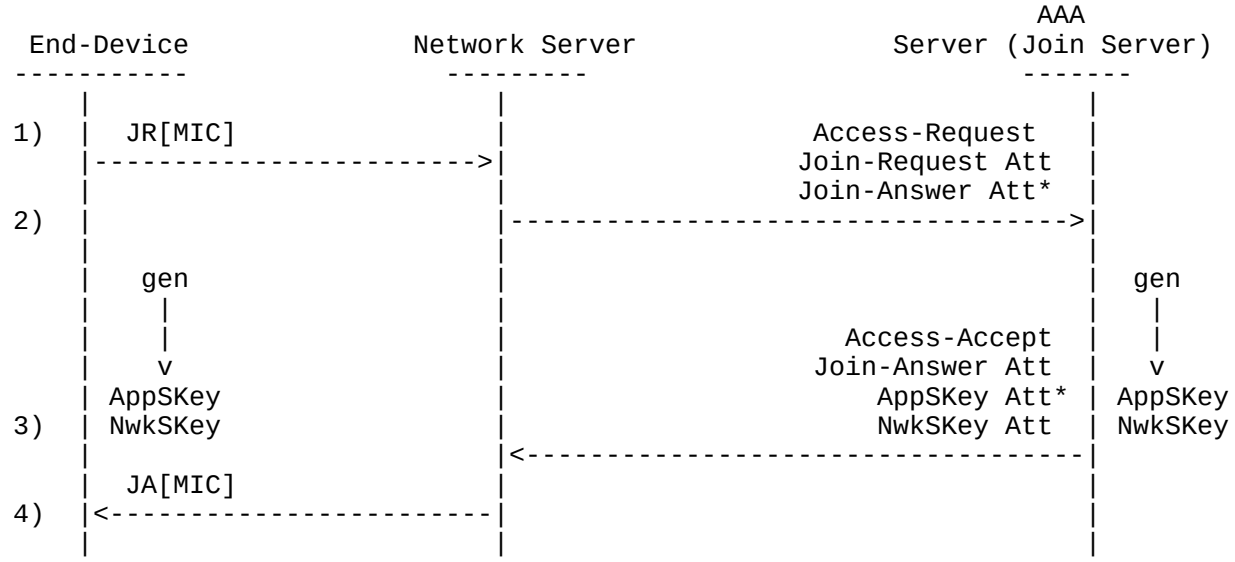
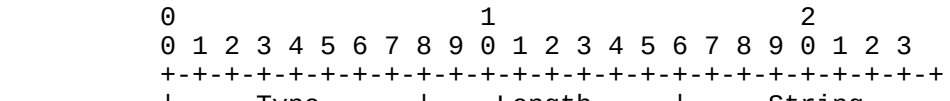


Figure 5: Protocol

4.3.1. Join-Request Attribute

Description

This Attribute contains the original Join-Request message. This attribute will only appear in the Access-Request message. A summary of the Join-Request attribute format is shown below. The fields are transmitted from left to right.



Type

TBD. for Join-Request

Length

18

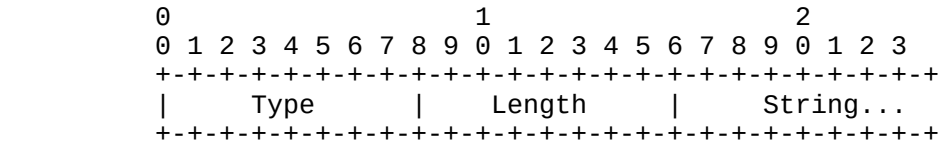
String

The String field contains an octet string with the Join-Request message as received over the network, such as defined in [LoRaWAN].

4.3.2. Join-Answer Attribute

Description

This Attribute is used in both RADIUS Access-Request and RADIUS Access-Accept messages. In the first case, it contains the Join Answer message with all the needed values filled by the network-server except the MIC (this fact is marked with an *). With these values, the Join Server (AAA server) that holds the AppKey is able to create the MIC and compose the final Join Answer message. In the second case, it contains the Join Answer with the MIC generated by the Join Server (AAA server). A summary of the Join-Answer attribute format is shown below. The fields are transmitted from left to right.



Type

TBD. for Join-Answer

Length

28

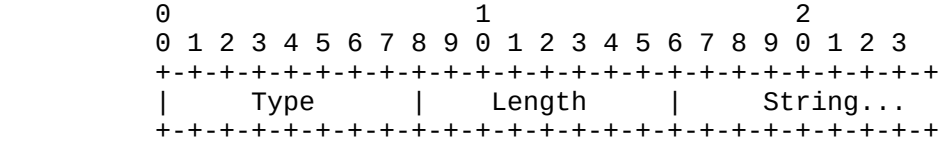
String

The String field contains an octet string with the Join-Answer as received over the network , as defined in [LoRaWAN].

4.3.3. AppSKey Attribute

Description

This Attribute contains the AppSKey, an application session key specific for the End-Device. This attribute is optional, and will only appear in the RADIUS Access-Accept message. A summary of the AppSKey attribute format is shown below. The fields are transmitted from left to right.



Type

TBD. for AppSKey

Length

16+

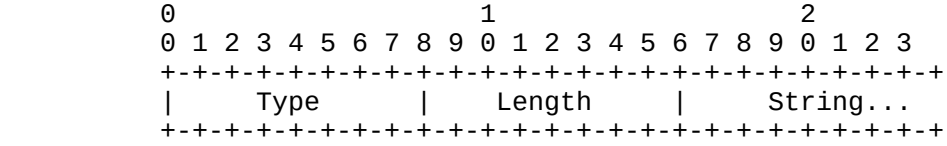
String

The String field contains an octet string containing the Application Session Key, as defined in [LoRaWAN].

4.3.4. NwkSKey Attribute

Description

This Attribute contains the NwkSKey, an network session key specific for the End-Device. This attribute will only appear in the Access-Accept message. A summary of the NwkSKey attribute format is shown below. The fields are transmitted from left to right.



Type

TBD. for NwkSKey

Length

16+

String

The String field contains the octet string of the Network Session Key , as defined in [LoRaWAN].

4.3.5. Table of Attribute

Request	Accept	Reject	Challenge	#	Attribute
1	0	0	0	TBD.	Join-Request
1	1	0	0	TBD.	Join-Answer
0	0-1	0	0	TBD.	AppSKey
0	1	0	0	TBD.	NwkSKey
Request	Accept	Reject	Challenge	#	Attribute

Figure 6: Attributes Table

5. Open Issues

With the purpose of extending the authentication process via AAA infrastructures, and concretely, RADIUS, we have faced a question regarding the relationship between the AppEUI associated to the organization operating the Join Server and the realm used by RADIUS to route the AAA information to the AAA Server (Join Server) of that organization.

In particular, the Network Server knows the AppEUI included in the Join Request, but it needs to discover the realm (Fully Qualified Domain Name) that corresponds to that organizations ID to be able to communicate with the concrete RADIUS server.

NOTE: One option MAY be to use the DNS system to provide the FQDN associated to an AppEUI (which is an EUI64 address). The mapping using DNS to find out the domain name associated to an EUI64 address has been described in [RFC7043]. However, we would need the inverse process. Nevertheless, this needs further discussion.

6. Security Considerations

In the LoRaWAN 1.0 specification, the AppSKey and NwkSKey are not sent over the network, they are derived in each of the endpoints that communicate, namely the End-Device and the Network Server. In this document we propose relegating the responsibility of deriving the Network Session Key and Application Session Key to the RADIUS server (the Join Server). These session keys need to be sent to the Network Server and if necessary to the application server.

To send the messages over the network between the RADIUS server and the RADIUS client (in this case the Network Server). How to provide confidentiality to the key distributed is outside the scope of this document, nevertheless RadSec (RFC6614) or extensions such as those defined in RFC 6218 may be considered to protect the distribution.

The AAA framework and its key management features become increasingly important as the use case of LoRaWAN adds functionality and complexity. This is the case for having the Application Server and Network Server as separate entities and each receive its keys. Although the utility is apparent in that specific case, it has to be considered in any other future use-case that may require key management and key distribution. Another point in favor of using AAA can be also appreciated since the modifications required by this proposal does not imply the modification of the protocols of the constrained link, but the unrestricted network that is used to manage LP-WAN.

7. Proof of concept implementation

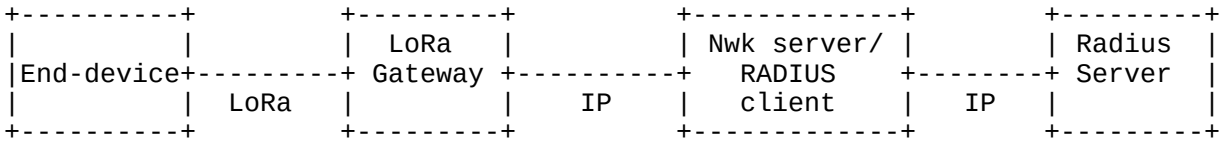
The proof of concept is implemented using the Go programming language, that is well suited for the development of web servers or a network servers as in this case.

The implementation of the network server is from [LoRaSERVER] which is tailored with the features of a RADIUS Client and the RADIUS server implementation from [RADIUSGo] that is modified to handle LoRaWAN attributes.

The LoRa end-device, pre-configured with AppKey, from Nemeus [MK002] is a USB key that can be controlled by UART (AT command) through USB interface. A JAVA application installed on a Linux machine is used to send control and data messages from the End-Device.

The LoRa Gateway is from EXPEMB [EXPEMB] which uses the packet forwarder to forward the LoRa packets to the LoRa Network Server. The Network Server is run in a docker container on a Linux machine transfers the LoRa packets into the RADIUS attributes to be sent to the RADIUS server. For now, the packets are sent to the default RADIUS server but in the future this would be changed as per the discussion in Section 5 in order to redirect the RADIUS request to appropriate RADIUS server.

The RADIUS server is run in a docker container on a Linux machine which contains the mapping between the DevEUI of the End-Device and the AppKey. This AppKey from the map along with the received LoRa attributes is used to derive the session keys, NwkSKey and AppSKey, in the RADIUS server. These keys are transported as RADIUS attributes back to the network server.



A successful authentication would result in the session keys, NwkSKey and AppSKey, being visible on the network server that can be viewed using a web interface and the DevAddr being acquired by the End-Device from the Join Accept Lora message. Running Wireshark on the interface between RADIUS server and the Network Server shows the RADIUS packets with the LoRa attributes.

To simplify the design and implementation, we opted for creating one RADIUS Attribute per message, instead of per each field within the message since only the authenticating module responsible for the Join Procedure in the current network server is delegated to the AAA server and the AAA server would be able to obtain the required fields from this single attribute, i.e either JoinRequest or JoinAccept message. This design choice would follow the RADIUS guidelines given in [RFC6158] identifying it as string for being an opaque encapsulation of data structures defined outside RADIUS. Creating an attribute per field, would be useful in case the AAA infrastructure would change its behavior depending on the specific content of one or more of the fields contained in the message. This could be the case when the LoRaWAN use case becomes more complex and add more functionality.

As future work, we intend to implement the proof of concept in FreeRADIUS

8. Acknowledgments

This work has been possible partially by the SMARTIE project (FP7-SMARTIE-609062 EU Project) and the Spanish National Project CICYT EDISON (TIN2014-52099-R) granted by the Ministry of Economy and Competitiveness of Spain (including ERDF support).

We also wanted to thank for the comments received about this document by Sri Gundavelli, Yeoh Chun-Yeow, Alan DeKok, Stephen Farrell and Mark Grayson.

9. IANA Considerations

In this document we define 4 new RADIUS Attributes that would need actions from IANA to assign the corresponding numbers.

Number	Name	Reference
TBD	Join-Request	Section 4 of this document
TBD	Join-Answer	Section 4 of this document
TBD	AppSKey	Section 4 of this document
TBD	NwkSKey	Section 4 of this document

10. References

10.1. 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, DOI 10.17487/RFC2865, June 2000, <http://www.rfc-editor.org/info/rfc2865>.

[RFC6158] DeKok, A., Ed. and G. Weber, "RADIUS Design Guidelines", BCP 158, RFC 6158, DOI 10.17487/RFC6158, March 2011, <http://www.rfc-editor.org/info/rfc6158>.

[RFC7043] Abley, J., "Resource Records for EUI-48 and EUI-64 Addresses in the DNS", RFC 7043, DOI 10.17487/RFC7043, October 2013, <http://www.rfc-editor.org/info/rfc7043>.

10.2. Informative References

[EXPEMB] EXPEMB, E., "LoRa MultiConnectivity Service Gateway - Last Accessed:", July 2016, <www.expemb.com/en/product/multi%E2%80%90connectivity-service-gateway-sgwmc%E2%80%90x86lr%E2%80%9012132/>.

[LoRaAllianceSecurity] Girard, P., "LoRaWAN - SECURITY a comprehensive insight - Online Resource: Last Accessed", July 2016, <http://portal.lora-alliance.org/DesktopModules/Inventures_Document/FileDownload.aspx?ContentID=1085>.

[LoRaSERVER] Acklio, A., "LoRa Server", July 2016, <http://www.ackl.io>.

[LoRaWAN] Sornin, N., Luis, M., Eirich, T., and T. Kramp, "LoRa Specification V1.0", January 2015, <https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf>.

[MK002] Nemesus, N., "MK002-xx-EU - Last Accessed:", July 2016, <http://www.nemeus.fr/en/mk002-usb-key>.

[RADIUSGo]
bronze1man, B., "Radius: A golang radius library - Last Accessed:", July 2016, <https://github.com/bronze1man/radius>.

Authors' Addresses

Dan Garcia-Carrillo (Ed.)
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

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

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

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

Arunprabhu Kandasamy
Acklio
2bis rue de la Chataigneraie
35510 Cesson-Sevigne Cedex
France

Email: arun@ackl.io

Alexander Pelov
Acklio
2bis rue de la Chataigneraie
35510 Cesson-Sevigne Cedex
France

Email: a@ackl.io

Network Working Group
INTERNET-DRAFT
Updates: 5176
Category: Standards Track
<draft-ietf-radext-coa-proxy-01.txt>
8 July 2016

DeKok, Alan
FreeRADIUS
J. Korhonen
Nokia Siemens Networks

Dynamic Authorization Proxying in Remote Authorization Dial-In User Service Protocol (RADIUS) draft-ietf-radext-coa-proxy-01.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/1id-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 February 8, 2016.

Copyright Notice

Table of Contents

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 ommission means that proxying of CoA packets is, in practice, impossible.

We correct that ommission here by explaining how an existing RADIUS attribute, Operator-Name (Section 4.1 of [RFC5580]), can be used to record the visited network for a particular session. We then explain how that attribute can be used by CoA proxies to route packets "backwards" through a RADIUS proxy chain. We introduce a new attribute; Operator-NAS-Identifier, and show how this attribute can increase privacy about the internal implementation of the visited netowkr.

We conclude with a discussion of the security implications of the design, and show how they are acceptable.

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) 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 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, and can ask the Home Network is the user is authentic (or not).

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 [RFC7542]. Other methods are possible, but we restrict ourselves to this usage, as it is the most common one.

The proxy 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 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.

If the "next hop" is a proxy, it will perform the same Realm lookup, and then proxy the packet. Alternatively, if the "next hop" is the Home Server for that realm, it will try to authenticate the user, and respond with an Access-Accept, Access-Reject, or Access-Challenge.

The RADIUS client will match the response packet to an outstanding request. If the client is part of a proxy, it will then proxy that response packet in turn to the system which 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. Once a reply has been recieved by the proxy, it can discard all information about the user.

The same proxy method is used for 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 the same as, the RADIUS server. Similarly, the CoA server is a system that is either co-located with, or the same as, the RADIUS client.

In the case of packets sent inside of one network, the source and

destination of CoA packets is 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 multiple networks are involved. [RFC5176] suggests that CoA proxying is permitted, but makes no suggestions for how it should be done.

If proxies tracked user sessions, it might be possible for a proxy to match an incoming CoA-Request to a user session, and then to proxy that packet to the RADIUS client which originated the Access-Request for that sessions.

There are many problems with such a scenario. The CoA server may, in fact, not be co-located with the RADIUS client. The RADIUS client may be down, but there may be a different CoA server which could accept the packet. User session tracking can be expensive and complicated for a proxy, and many proxies do not record user sessions. Finally, [RFC5176] is silent on the topic of "session identification attributes", which makes it impossible for a proxy to determine if a CoA packet matches a particular user session.

The result is that CoA proxying cannot be performed when using the behavior defined in [RFC5176].

3. How to Perform CoA Proxying

The solution to the above problem is to use 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.

3.1. Operator-Name in Access-Request and Accounting-Request packets

When a Visited Network proxies an Access-Request or Accounting-Request packet outside of its network, it 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, one should be chosen, and used for all packets.

Visited Networks MUST use a consistent value for Operator-Name for 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.

Proxies which record user session information SHOULD also record Operator-Name. Proxies which do not record user session information SHOULD NOT record Operator-Name.

Home Networks SHOULD record Operator-Name along with other information about user sessions. Home Networks which expect to send CoA packets to Visited Networks MUST record Operator-Name for each user session which originates from a Visited Network.

Networks which contain both the RADIUS client and RADIUS server do not need to record or track Operator-Name.

3.2. Operator-Name in 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 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. In this case, the destination of the packet is not a RADIUS server, but a CoA server.

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

Proxies which receive the CoA packet MUST look up the realm from the Operator-Name in a logical "realm routing table", as with Home Servers, above. This process continues with any additional proxies until the packet reaches the Visited Network.

The Visited Network can then send the CoA packet to the NAS, and return any response packet back up the proxy chain to the Home Server.

Networks which contain both the CoA client and CoA server do not need to record or track Operator-Name.

3.3. Operator-NAS-Identifier

The process described in the previous section allows for CoA proxying, but it does not support privacy for Visited Networks. That is, all "internal" information about the Visited Network is public. This information includes NAS-Identifier, NAS-IP-Address, NAS-IPv6-Address, etc. We belive that the internals of the Visited Network should be opaque to third parties.

In addition, we will see that privacy provisions can have a positive impact on the security of the system.

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

An Operator-NAS-Identifer attribute SHOULD be added to an Access-Request or Accounting-Request packet by a Visited Network just before proxying a packet to an external RADIUS server. When the Operator-NAS-Identifer attribute is added to a packet, 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. That is, for everyone outside of the Visited Network, the identity of the NAS is the Visited Network. For the Visited Network, the identity of the NAS is private information, which is opaque to everyone else.

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. Requirements

4.1. Requirements on Home Servers

A Home Server MUST NOT send CoA packets for users who are not part of its realm. The provisions of the next few sections describe how other participants in the RADIUS ecosystem can enforce this requirement.

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 any Operator-NAS-Identifier it has recorded for that session.

4.2. Requirements on Visited Networks

A Visited Network which receives a proxied CoA packet MUST perform all of the checks discussed above for proxies. 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.

Due to the requirements of Section 2.3 of [RFC5176], a Visited Network MUST remove Operator-Name and Operator-NAS-Identifier from any CoA-Request or Disconnect-Request packet prior to proxying that packet to a CoA server. This requirement is phrase more generically below, in Section XX.

While a Visited Network may create an Operator-NAS-Identifier via many methods. The value SHOULD be cryptographically strong. It SHOULD be verifiable by the Visited Network, without tracking every single user session.

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 that these systems are co-located.

4.3.1. Security Requirements on Proxies

Section 6.1 of [RFC5176] has some security requirements on proxies which 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 change that requirement to a proxy MUST perform a "reverse path forwarding" (RPF) check to verify that a Disconnect-Request or CoA-Request originates from an authorized Dynamic Authorization Client. Without this change, a proxy may forward forged packets, and thus contribute to the forgery problem instead of preventing it.

Proxies which 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

CoA-NAK or Disconnect-NAK packet, which contains an Error-Cause attribute having value 503 ("Session Context Not Found").

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

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 substantiall different from changing (or refusing to connect) a user session during the initial process of authentiction.

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.

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. Instead, we say that for a CoA proxy, all attributes MUST NOT be treated as mandatory. Proxies SHOULD perform proxying based on Operator-Name, but other schemes are possible (though not discussed here). Proxies SHOULD forward all packets as-is, with minimal changes. Only the final CoA server (i.e RADIUS NAS) can make a decision on which attributes are mandatory and which are not.

Where Operator-Realm and Operator-NAS-Identifier is received by a proxy, the proxy MUST pass those attributes through unchanged.

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 which travel the reverse path. The result is that a NAS will only ever receive CoA packets which contain either attributes sent by the NAS to the RADIUS server, or attributes which are required to perform a change of authorization.

We note that the above requirement applies not only to Operator-Name and Operator-NAS-Identifier, but also to any future attributes which are added by a RADIUS proxy.

In short, proxies SHOULD behave much like a CoA server, and where possible, perform many of the same validations done by a CoA server.

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 attempting authenticastion in a visited network. The login attempt is done via a visited NAS. 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 proxy the authentication request to an upstream server. That server may be the home server for the user, or it may be another proxy.

The visited RADIUS server should add an Operator-Name attribute, with value "1" followed by it’s own realm name. e.g. "1example.com". Where the visited network has multiple realms, it MUST choose a realm name which permits packets to be routed back to itself. The visited RADIUS server MAY also add an Operator-NAS-Identifier as discussed below.

The upstream proxy or proxies will then forward the packet to the home server. Intermediate proxies MUST NOT modify the contents of, or delete the Operator-Name or Operator-NAS-Identifier attributes.

The Home Server SHOULD record both Operator-Name and Operator-NAS-Identfier along with other information about the users session.

5.2. CoA Proxing

When the Home Server decides to disconnect a user, it looks up the Operator-Name and Operator-NAS-Identifer, 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 recieves 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.

6. 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.

7. IANA Considerations

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

8. References

8.1. 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.

[RFC7542]
DeKok A., "The Network Access Identifier", RFC 7542, May 2015.

[DATA]
DeKok A., "Data Types in the Remote Authentication Dial-In User Service Protocol (RADIUS)", draft-ietf-radext-datatypes-02.txt, November 2015

8.2. 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
Broadcom Corporation
3151 Zanker Road
San Jose, California 95134
United States

EMail: jouni.nospam@gmail.com

RADIUS Extensions Working Group
Internet-Draft
Updates: 3748 (if approved)
Intended status: Best Current Practice
Expires: January 9, 2017

S. Winter
RESTENA

July 08, 2016

Considerations regarding the correct use of EAP-Response/Identity
draft-ietf-radext-populating-eapidentity-01

Abstract

There are some subtle considerations for an EAP peer regarding the content of the EAP-Response/Identity packet when authenticating with EAP to an EAP server. This document describes two such considerations and suggests workarounds to the associated problems. One of these workarounds is a new requirement for EAP peers that the use of UTF-8 is required for the content of EAP-Response/Identity (which updates RFC3748).

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2017.

Copyright Notice

Table of Contents

1. Introduction

1.1. Problem Statement

An Extensible Authentication Protocol (EAP, [RFC3748]) conversation between an EAP peer and an EAP server starts with an (optional) request for identity information by the EAP server (EAP-Request/ Identity) followed by the peer's response with identity information (EAP-Response/Identity). Only after this identity exchange are EAP types negotiated.

EAP-Response/Identity is sent before EAP type negotiation takes place, but it is not independent of the later-negotiated EAP type. Two entanglements between EAP-Response/Identity and EAP methods' notions of a user identifier are described in this document.

1. The choice of identifier to send in EAP-Response/Identity may have detrimental effects on the subsequent EAP type negotiation.

2. Using identifiers from the preferred EAP type without thoughtful conversion of character encoding may have detrimental effects on the outcome of the authentication.

The following two chapters describe each of these issues in detail. The last chapter contains recommendations for implementers of EAP peers to avoid these issues.

1.2. Taxonomy of identities in EAP

The notion of identity occurs numerous times in the EAP protocol stack (EAP-Response/Identity, Outer identity, method-specific identity, tunneled identity). This document uses the following terminology when discussing EAP identities.

- User Identifier: Each EAP method has a means to identify the user or machine that tries to authenticate. There are no restrictions on the format or encoding of this identifier. The user identifier is often also referred to as "method-specific identity". If an EAP method distinguishes between the user identifier and a realm identifier (see next bullet), then the user identifier is also often referred to as the "inner/true/real identity".

- Realm Identifier: Some EAP methods allow privacy-preserving enhancements where a string is sent which is actually not necessarily related to the user or machine that tries to authenticate. This identifier is often also referred to as "outer identity" or "roaming identity" or "anonymous outer identity". There is often a relationship between the realm identifier and the user identifier (e.g. they often share the same NAI realm suffix); but this is not a requirement. There are no restrictions on the format or encoding of the realm identifier. Realm identifiers are either

  * explicitly configured (e.g. string input UI in EAP peer: "Outer Identity")

  * implicitly configured by copying the actual user identifier

  * implicitly configured by copying the NAI realm of the user identifier and prefixing it non-configurably with a fixed privacy-preserving local username part like "anonymous" or the empty string (see [RFC7542])

  * configured in a mixed way, e.g. using a explicit string input UI for the local part of the realm identifier and combining it implicitly with a copy of the NAI realm part of the user identifier

- EAP-Response/Identity: a string representing the user or machine that tries to authenticate, used outside the EAP method-specific context for the entire EAP conversation. There can be only one EAP-Response/Identity per EAP conversation, even if that conversation could negotiate more than one EAP method to authenticate with. As per [RFC3748] there is no encoding requirement on EAP-Response/Identity (which this document changes:

the encoding MUST be UTF-8). In AAA protocol routing contexts, the content of EAP-Response/Identity is often used for request routing purposes. EAP-Response/Identity is chosen from the set:

* all realm identifiers from all configured EAP types supporting the notion of a realm identifier

* all user identifiers from all configured EAP types without the notion of a realm identifier

Several EAP types in a local configuration may share the same user and/or realm identifiers. The set of identifiers for EAP-Response /Identity may thus contain fewer elements than there are configured EAP types in a local configuration. One of the two problems addressed in this document stems from this fact: the set of identifiers may contain more than one element. The resulting EAP-Response/Identity always routes all configured EAP types to only one destination, even if different EAP types would need routing to different destinations.

o User-Name: when using EAP in AAA protocol contexts (e.g. RADIUS [RFC2865], Diameter [RFC6733]), this additional identifier is created outside the EAP peer (typically in a pass-through authenticator) by copying EAP-Response/Identity content to the AAA protocol's User-Name attribute. There is no format requirement on User-Name, but there is an encoding requirement: the string MUST be UTF-8 encoded. One of the two problems addressed in this document stems from this fact: EAP-Response/Identity does not have an encoding requirement, nor does it carry meta-information about the encoding used - and yet, it needs to be coerced into a UTF-8 encoding.

o Further identifiers: Some EAP methods establish an EAP session inside EAP (e.g. PEAP first establishes a TLS tunnel using a realm identifier, and then starts an EAP exchange inside the tunnel). This being a new, independent EAP session, it contains its own EAP-Response/Identity, can invoke EAP method negotiation with different (inner) EAP types (this happens e.g. with EAP-FAST and its configurable choice of EAP-GTC or EAP-MSCHAPv2 inside the inner EAP session), and those inner EAP methods then have their own user identifiers. Where the inner EAP method itself supports the notion of realm identifiers, another identifier could be configured. For the purposes of this document, none of those details are considered and the process by which the (outer) EAP method selects its user identifier is left entirely to that EAP type. This document does not consider the (inner) EAP-Response/ Identity in scope; the recommendations in this document to not apply to such (inner) occurences of EAP-Response/Identity.

1.3. 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 RFC 2119. [RFC2119]

2. EAP-Response/Identity: Effects on EAP type negotiation

Assuming the EAP peer's EAP type selection is not the trivial case (i.e. it has more than one configured EAP type for a given network or application, and needs to make a decision which one to use), an issue arises when the configured EAP types are not all configured with the same realm identifier (or user identifier for EAP types not supporting the notion of a realm identifier).

Issue: if the identifiers in the set of configured EAP types differ (e.g. have a different [RFC7542] "realm" portion), and the authenticator does not send identity selection hints as per [RFC7542], then EAP type negotiation may be limited to those EAP types which are terminated in the same EAP server. The reason for that is because the information in the EAP-Response/Identity is used for request routing decisions and thus determines the EAP server - a given realm identifier may be routed to a server which exclusively serves the corresponding EAP types. Negotiating another EAP type from the set of configured EAP types during the running EAP conversation is then not possible.

Example:

Assume an EAP peer is configured to support two EAP types:

- EAP-AKA' [RFC5448] with user identifier imsi@mnc123.mcc123.3gppnetwork.org; the configuration is set up to authenticate only to
  - cellular networks
  - Wi-Fi Passpoint networks which advertise support for the MNC 123 and MCC 123

  The EAP server for this EAP type is in a host under control of the 3GPP consortium

- EAP-TTLS [RFC5281] with user identifier "john@realm.example" and realm identifier "@realm.example"; the configuration is set up to authenticate only to

* Wi-Fi networks with the SSID "eduroam"

* Wi-Fi Passpoint networks which advertise support for the roaming consortium 00-1B-C5-04-60 (the eduroam consortium)

* wired ethernet

The EAP server for this EAP type is in a host under control of the eduroam consortium

The user approaches a Passpoint Wi-Fi hotspot with SSID "arbitrary" which emits a beacon advertising support for the MNC 123/MCC 123 AND for the consortium identifier 00-1B-C5-04-60. The local configuration thus yields two different EAP type candidates for authentication to the network. Unbeknownst to the user's device, the credit with the 3G provider is fully depleted and the user will be unable to authenticate with his EAP-AKA' credentials. Using his identifier of the roaming consortium eduroam (see also [RFC7593]), he could authenticate with EAP-TTLS and his john@realm.example user identifier. Identity selection hints are not sent.

Consequence: If the EAP peer consistently chooses the imsi@mnc123.mcc123.3gpp-network.org user identifier as choice for its initial EAP-Response/Identity, requests will always be routed to the 3GPP consortium EAP server, and the user will be consistently and perpetually rejected, even though in possession of a valid credential for the hotspot.

An EAP peer should always try all options to authenticate. As the example above shows, it may not be sufficient to rely on EAP method negotiation alone to iterate through all configured EAP types and come to a conclusive outcome of the authentication attempt. Multiple new EAP authentications, each using an EAP-Response/Identity from a different element of the set of realm identifiers, may be required to fully iterate through the list of usable identities.

3. Character (re-)encoding may be required

The user identifiers as configured in the EAP method configuration are not always suited as realm identifiers to choose as EAP-Response/Identity: EAP methods define the encoding of their method-specific outer identities at their leisure; in particular, the chosen encoding may or may not be UTF-8.

It is not the intention of EAP, as a mere method-agnostic container which simply carries EAP types, to restrict an EAP method's choice of encoding of user identifiers. However, there are restrictions in what should be contained in the EAP-Response/Identity: EAP is very

often carried over a AAA protocol (e.g over RADIUS as per [RFC3579]). The typical use for the contents of EAP-Response/Identity inside AAA protocols like RADIUS [RFC2865] and Diameter [RFC6733] is to copy the content of EAP-Response/Identity into a "User-Name" attribute; the encoding of the User-Name attribute is required to be UTF-8. EAP-Response/Identity does not carry encoding information itself, so a conversion between a non-UTF-8 encoding and UTF-8 is not possible for the AAA entity doing the EAP-Response/Identity to User-Name copying.

Consequence: If an EAP method's user identifier is not encoded in UTF-8, and if the EAP peer verbatimly uses that user identifier for its EAP-Response/Identity field, then the AAA entity is forced to violate its own specification because it has to, but can not use UTF-8 for its own User-Name attribute. If the EAP method supports a separate realm identifier in a non UTF-8 character set, and the EAP peer verbatimly uses that realm identifier for its EAP-Response/Identity field, then the same violation occurs.

This jeopardizes the subsequent EAP authentication as a whole; request routing may fail, lead to a wrong destination or introduce routing loops due to differing interpretations of the User-Name in EAP pass-through authenticators and AAA proxies.

4. Recommendations for EAP peer implementations

Where realm identifiers or user identifiers between multiple configured EAP types in an EAP peer differ, the EAP peer can not rely on the EAP type negotiation mechanism alone to provide useful results. If an EAP authentication gets rejected, the EAP peer SHOULD re-try the authentication using a different EAP-Response/Identity than before. The EAP peer SHOULD try all possible EAP-Response/Identity contents from the entire set of configured EAP types before declaring final authentication failure.

EAP peers need to maintain state on the encoding of the configured user identifiers and realm identifiers which are used in their local EAP type configuration. When constructing an EAP-Response/Identity from the set of available identifiers, they MUST (re-)encode the corresponding identifier as UTF-8 and use the resulting value for the EAP-Response/Identity.

Where an EAP method supports privacy-preserving realm identifiers, those SHOULD be configured for user privacy reasons. For deployments of such EAP types, these realm identifiers MUST be in the the format Network Access Identifier (NAI), see [RFC7542] if the realm identifiers are expected to become used beyond the scope of a single, closed enterprise. Even in such closed environments, the NAI format is RECOMMENDED. The RECOMMENDED format for the local part of the

realm identifier is the empty string; where this is not possible the suggested alternative is the string "anonymous".

5. Privacy Considerations

Because the EAP-Response/Identity content is not encrypted, the backtracking to a new EAP-Response/Identity will systematically reveal all configured identifiers to intermediate passive listeners on the path between the EAP peer and the EAP server (until one authentication round succeeds).

This additional leakage of identity information is not very significant though, because where privacy is considered important, the additional option for separate privacy-preserving realm identifiers which is present in most modern EAP methods can and should be used.

If the EAP peer implementation is certain that all EAP types will be terminated at the same EAP server (e.g. with a corresponding configuration option) then the iteration over all identities can be avoided, because EAP type negotiation is then sufficient.

If a choice of which identity information to disclose needs to be made by the EAP peer, when iterating through the list of identifiers the EAP peer SHOULD

- in first priority honour a manually configured order of preference of EAP types, if any
- in second priority try EAP types in order of less leakage first; that is, EAP types with a privacy-preserving realm identifier that differs from the user identifier should be tried before other EAP types which would reveal the corresponding actual user identifiers.

6. Security Considerations

The security of an EAP conversation is determined by the EAP method which is used to authenticate. This document does not change the actual authentication with an EAP method, and all the security properties of the chosen EAP method remain. The format requirements (character encoding) and operational considerations (re-try EAP with a different EAP-Response/Identity) do not lead to new or different security properties.

7. IANA Considerations

There are no IANA actions in this document.

8. References

8.1. Normative References

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

8.2. Informative References

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

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

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC5281] Funk, P. and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, August 2008.

[RFC5448] Arkko, J., Lehtovirta, V., and P. Eronen, "Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')", RFC 5448, May 2009.

[RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", RFC 6733, October 2012.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <http://www.rfc-editor.org/info/rfc7542>.

[RFC7593] Wierenga, K., Winter, S., and T. Wolniewicz, "The eduroam Architecture for Network Roaming", RFC 7593, DOI 10.17487/RFC7593, September 2015, <http://www.rfc-editor.org/info/rfc7593>.

Author's Address

Stefan Winter
Fondation RESTENA
6, rue Richard Coudenhove-Kalergi
Luxembourg 1359
LUXEMBOURG

Phone: +352 424409 1
Fax: +352 422473
EMail: stefan.winter@restena.lu
URI: http://www.restena.lu.